

# TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

## PROGRAMACIÓN II

### TRABAJO FINAL INTEGRADOR

Sistema de Gestión de Usuarios y Credenciales de Acceso

Integrantes:

**Barandiarán, Francisco**

**De Inocenti, Alfredo**

**Olivera, Favio**

**Rao, Maximiliano**

---

### 1. Equipo y roles

El proyecto fue desarrollado colaborativamente por un equipo de 4 integrantes, distribuyendo las responsabilidades de la siguiente manera:

Integrante	Rol Principal	Responsabilidades
<b>Barandiarán, Francisco</b>	Arquitecto de Datos	Modelo físico - scripts SQL - soporte a DAO
<b>De Inocenti, Alfredo</b>	Desarrollador Backend	Lógica de negocio - Transacciones - <i>AbstractService</i>
<b>Olivera, Favio</b>	Desarrollador de Entidades	Modelado - Entidades ( <i>Entities</i> ) - Relación 1→1 - Diagrama UML
<b>Rao, Maximiliano</b>	Desarrollador Frontend	Presentación/UX consola - Menú ( <i>Main</i> ) - Flujos - Validaciones de entrada

### 2. Dominio y justificación

**Dominio:** Gestión de usuarios y credenciales (base de cualquier login/registro).

**Motivos:**

- Relevancia práctica (web, mobile, empresariales).
- Complejidad didáctica adecuada: relación 1→1, soft delete, transacciones, validaciones (unicidad).
- Enfoque en seguridad (hash + salt, reset, auditoría básica).
- Escalable a roles/permisos, sesiones, OAuth/JWT.

**Casos de uso:** alta de usuario con credencial, autenticación (validación), cambio de contraseña, activar/desactivar, eliminación lógica, consultas de auditoría.

**Alcance acotado:** sin cifrado real (hash simulado), sin tokens/sesiones, sin roles ni recuperación por email.

---

### 3. Decisiones de Diseño

#### 3.1 Relación 1→1 Usuario–CredencialAcceso (FK única)

- **Racional:** 1 usuario ↔ 1 credencial; tablas independientes; orden natural de creación (credencial→usuario); soft delete por separado; lectura legible mediante FK única (no PK compartida).<br>
- **Trade-off:** requiere JOIN para vistas completas.

#### Esquema (resumen)

sql

CredencialAcceso(id PK, hashPassword, salt, ultimoCambio, requiereReset, eliminado)  
Usuarios(id PK, username **UNIQUE**, email **UNIQUE**, activo, fechaRegistro, credencial  
**UNIQUE** FK→CredencialAcceso(id), eliminado)

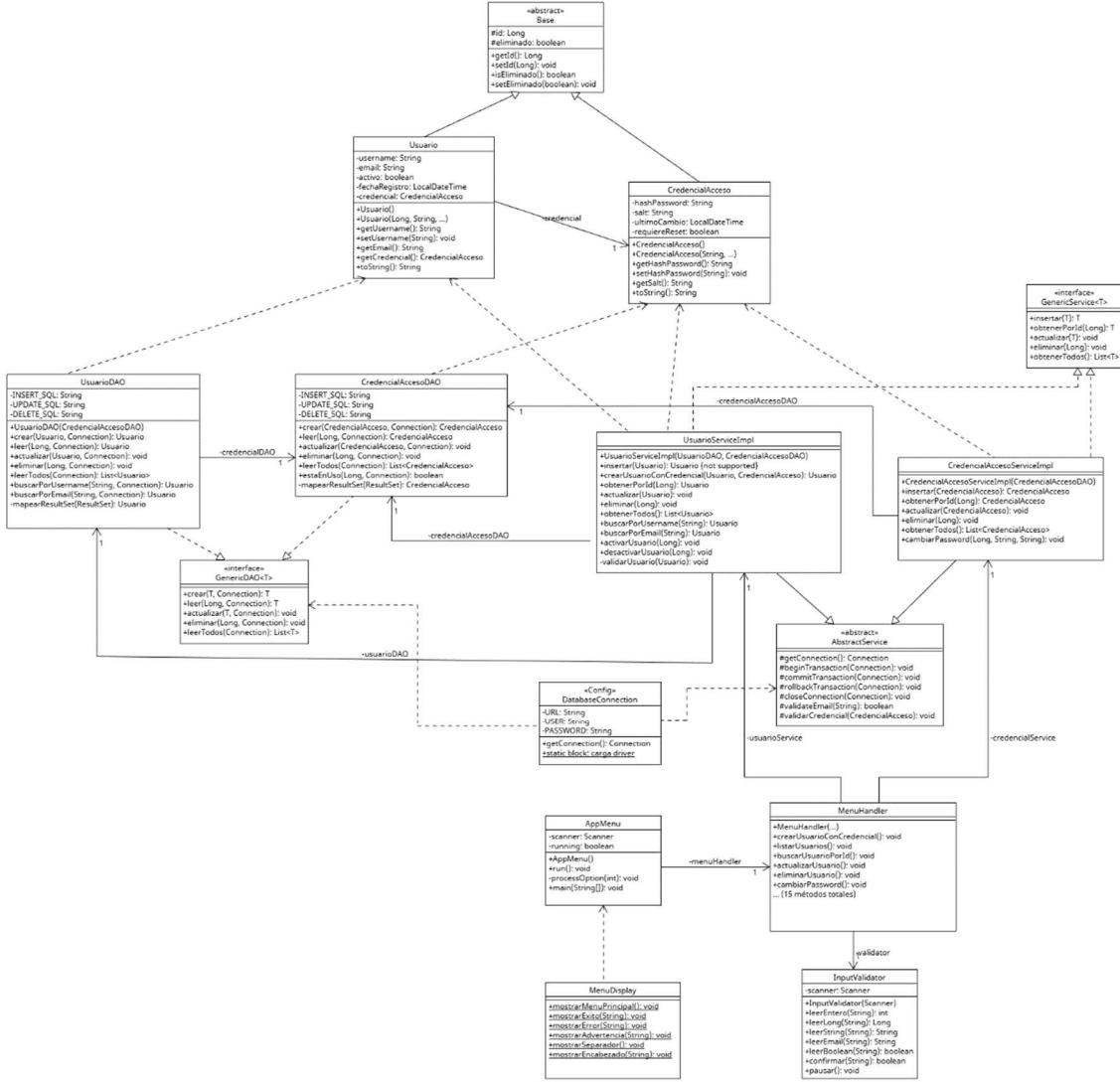
---

#### 3.2 Soft delete (vs hard delete)

- **Elegido:** *eliminado=TRUE* en Usuario y Credencial (preserva histórico, permite auditoría, evita cascadas destructivas).
- **Base común:** clase *Base { id, eliminado }*.

#### 3.3 Diagrama UML del Modelo

Usuario-CredencialAcceso:



### 3.4 Validaciones de datos (resumen)

- *username*: único, ≤30, alfanumérico/\_
- *email*: único, ≤120, formato válido
- *hashPassword*: ≤255
- *salt*: ≤64
- Regla 1→1 obligatoria (usuario siempre con credencial válida).

### 3.5 Orden de Operaciones Crítico

**Crear usuario:** (1) crear credencial → (2) asignar FK → (3) crear usuario → (4) commit.

**Eliminar usuario:** marcar eliminado en Usuario y en su Credencial (misma transacción).

## 4. Arquitectura del Sistema

### Representación Gráfica

- Interacción con usuario
  - Validación de formato de entrada
- 
- 



- 
- 
- CAPA 3: LÓGICA DE NEGOCIO (Service)
- Validaciones de negocio
  - Gestión de transacciones (commit/rollback)
- 
- 



- 
- 
- CAPA 2: ACCESO A DATOS (DAO)
- Ejecución de queries SQL
  - Mapeo ResultSet → Objetos
- 
- 



- 
- 
- CAPA 1: MODELOS (Entities)
- Representación de entidades del dominio
  - Relaciones entre objetos
- 
- 



MySQL Database

---

## 5. Gestión de Persistencia y transacciones

### Restricciones implementadas:

- *UNIQUE* en username, email, credencial (garantiza 1→1)
- *NOT NULL* en campos obligatorios
- *DEFAULT* para campos con valores iniciales
- *ON DELETE CASCADE* NO usado (soft delete manual)

**Transacionalidad (patrón):** begin → operaciones (misma conexión) → commit; ante error → rollback.

**Seguridad SQL:** todo con PreparedStatement (evita inyecciones).<br>

**Conexión:** DatabaseConnection carga el driver y expone getConnection(); try-with-resources en operaciones no transaccionales.

---

## 6. Reglas de Negocio

- **RN-001: Unicidad de Username**
- **RN-002: Unicidad de Email**
- **RN-003: Relación 1→1 Obligatoria**
- **RN-004: No Eliminar Credencial en Uso**
- **RN-005: Cambio de Contraseña Resetea Flag**
- **RN-006: Soft Delete en Cascada**

---

## 7. Pruebas y Verificación

### Menú Principal

```
□□□□□ SISTEMA DE GESTIÓN DE USUARIOS □□□□□
□□□□□ GESTIÓN DE USUARIOS □□□□□
□ 1. Crear Usuario con Credencial □
□ 2. Listar todos los Usuarios □
□ 3. Buscar Usuario por ID □
□ 4. Buscar Usuario por Username □
□ 5. Buscar Usuario por Email □
□ 6. Actualizar Usuario □
□ 7. Eliminar Usuario □
□ 8. Activar Usuario □
□ 9. Desactivar Usuario □
□□□□□ GESTIÓN DE CREDENCIALES □□□□□
□ 10. Crear Credencial (independiente) □
□ 11. Listar todas las Credenciales □
□ 12. Buscar Credencial por ID □
□ 13. Actualizar Credencial □
□ 14. Eliminar Credencial □
□ 15. Cambiar Password de Credencial □
□□□□□ SISTEMA □□□□□
□ 0. Salir □
□□□□□ Ingrese una opción:
```

### Operación Exitosa (Crear Usuario)

```
□□□□□ CREAR USUARIO □□□□□
Username: testuser2025
Email: test2025@example.com
¿Usuario activo? (S/N): s

--- Credencial de Acceso ---
Hash de contraseña: 5f4d87982fbcd3765d61da94a8fec4f
Salt: abc1test_2025
¿Requiere reset de contraseña? (S/N): n

□ ÉXITO: Usuario creado exitosamente con ID: 4
Usuario{id=4, username='testuser2025', email='test2025@example.com', activo=true, fechaRegistro=2025-11-15T08:45:26.483856200, tieneCredencial=true, credencialId=4, eliminado=false}

Presione Enter para continuar...]
```

### Manejo de Error (Error de validación)

```
█████████████████████████████████████████████████████████████████████████████████████
    CREAR USUARIO
█████████████████████████████████████████████████████████████████████████████████████
Username: testuser2025
Email: otro@example.com
¿Usuario activo? (S/N): s

--- Credencial de Acceso ---
Hash de contraseña: hash123
Salt: salt123
¿Requiere reset de contraseña? (S/N): n

□ ERROR: No se pudo crear el usuario: Error al crear usuario con credencial: El username ya existe: testuser2025
Presione Enter para continuar...|
```

## Listado de Usuarios (Lista completa)

```
█████████████████████████████████████████████████████████████████████████████████████
    LISTA DE USUARIOS
█████████████████████████████████████████████████████████████████████████████████████

Total de usuarios: 4
█████████████████████████████████████████████████████████████████████████████████████

ID: 1
Username: juanperez
Email: juanperez@example.com
Activo: Sí
Fecha Registro: 2025-11-15T08:18:09
Credencial ID: 1

█████████████████████████████████████████████████████████████████████████████████████

ID: 2
Username: maria_lopez
Email: maria.lopez@example.com
Activo: No
Fecha Registro: 2025-11-15T08:18:09
Credencial ID: 2

█████████████████████████████████████████████████████████████████████████████████████

ID: 3
Username: admin_user
Email: admin@example.com
Activo: Sí
Fecha Registro: 2025-11-15T08:18:09
Credencial ID: 3

█████████████████████████████████████████████████████████████████████████████████████

ID: 4
Username: testuser2025
Email: test2025@example.com
Activo: Sí
Fecha Registro: 2025-11-15T08:45:26
Credencial ID: 4

█████████████████████████████████████████████████████████████████████████████████████

Presione Enter para continuar...|
```

## Soft delete – (antes/después):

Antes

Query 1

```

1 •  SELECT id, username, eliminado
2   FROM Usuarios
3 WHERE id = 1;

```

Result Grid

	id	username	eliminado
▶	1	juanperez	0
*	NULL	NULL	NULL

### Eliminar desde app

```

ELIMINAR USUARIO
Ingrese ID del usuario a eliminar: 1

Usuario a eliminar:
Usuario{id=1, username='juanperez', email='juan.perez.nuevo@example.com', activo=true, fechaRegistro=2025-11-15T08:18:09,
tieneCredencial=true, credencialId=1, eliminado=false}

 ADVERTENCIA:
 ATENCIÓN: Esta operación también eliminará la credencial asociada.

¿Está seguro que desea eliminar este usuario? (S/N): s

 ÉXITO: Usuario eliminado correctamente (soft delete).

Presione Enter para continuar...

```

### Después

Query 1

```

1 •  SELECT id, username, eliminado
2   FROM Usuarios
3 WHERE id = 1;

```

Result Grid

	id	username	eliminado
▶*	1	juanperez	1
*	NULL	NULL	NULL

## 8. Conclusiones y Trabajo Futuro

- Logros:

- Arquitectura en 4 capas con bajo acoplamiento/alta cohesión.
- Transacciones ACID (rollback ante fallas).
- Relación 1→1 robusta con FK única y unicidades.
- Soft delete operativo y verificable.
- Mitigación de SQL injection con PreparedStatement.

### **✖ Limitaciones actuales:**

- Hash simulado (sin BCrypt/Argon2).
- Sin roles/permisos, ni recuperación por email.
- Sin pool de conexiones, ni paginación/listados grandes.
- Sin tests automatizados ni logging estructurado.

### **⚠ Mejoras propuestas:**

- BCrypt para passwords reales.
- Tests JUnit (duplicados, transacciones con rollback).
- Roles/Permisos (tablas puente).
- Recuperación por email (token temporal).
- Pool de conexiones (HikariCP) y logging (SLF4J).
- API REST (Spring Boot) como evolución natural de la capa Service.

---

## **9. Referencias y Recursos Utilizados**

### **9.1 Herramientas Utilizadas**

#### **Desarrollo:**

- **IDE:** NetBeans

#### **Base de Datos:**

- **Cliente:** MySQL Workbench

#### **Control de Versiones:**

- **Git:** Git
- **GitHub:** Repositorio del proyecto  
([https://github.com/MaximilianoRao/Grupo22\\_Integrador\\_Programacion\\_II](https://github.com/MaximilianoRao/Grupo22_Integrador_Programacion_II))

#### **Diagramación:**

- **UML:** UMLetino

#### **Documentación:**

- **Markdown:** Para README.md

### **9.2 Uso de Inteligencia Artificial**

### **Declaración de Transparencia:**

En el desarrollo de este proyecto se utilizó **Claude AI (Anthropic)** y **ChatGPT (OpenAI)**: como herramienta de asistencia para:

#### **1. Consultas técnicas específicas:**

- Sintaxis de JDBC
- Mejores prácticas de manejo de excepciones

#### **2. Revisión de código:**

- Detección de posibles SQL injection
- Sugerencias de optimización

#### **3. Documentación:**

- Redacción de este informe (organización y claridad)
- Ejemplos de código para explicaciones

### **Otras herramientas IA consultadas:**

- **GitHub Copilot:** Sugerencias de autocompletado (desactivado para lógica crítica)

## **9.3 Material del Curso**

### **Apuntes y Clases:**

- Material teórico de Programación II (Tecnicatura en Programación)
- Prácticos de JDBC y bases de datos relacionales
- Ejemplos de código proporcionados por el docente

## **9.4 Video**

Link video:

<https://www.youtube.com/watch?v=fz9zXKPfM2k>