## **Grupo-B-Proyecto-Integrador-I**

**DOCUMENTACIÓN DEL DISEÑO DE BASE DE DATOS**

## **REPOSITORIO: Programacion-BBDD**

## **URL:** <https://github.com/Grupo-B-Proyecto-Integrador-I/Programacion-BBDD.git>

**Profesores: Ivana Soledad ROJAS CÓRSICO**

**Augusto SCHAUMBURG**

**Alumnos:** Melina Belén Bruvera

Alvaro Fernando Galiño Velez

Flavia Guadalupe Sicchar Gomez

**ÍNDICE**

[**Grupo-B-Proyecto-Integrador-I 1**](#_flxh8c35b1ob)

[**REPOSITORIO: Programacion-BBDD 1**](#_ez0ykliyvok0)

[**URL: https://github.com/Grupo-B-Proyecto-Integrador-I/Programacion-BBDD.git 1**](#_g4pbyvrs1cxm)

[**1. INTRODUCCIÓN 3**](#_pru4wq4kbk2)

[**2. DIAGRAMA ENTIDAD-RELACIÓN (DER) 3**](#_fuypxemgwvz2)

[**3. MODELO RELACIONAL 4**](#_mfjx74oftcea)

[**Tabla: Rol 4**](#_l34rr7kgipr1)

[**Tabla: User 4**](#_oleo09i3gqoe)

[**4. DESCRIPCIÓN DE ENTIDADES 6**](#_jr6t95d1bdm4)

[**5. NORMALIZACIÓN DE LA BASE DE DATOS (HASTA LA TERCERA FORMA NORMAL - 3NF) 7**](#_wjka7iwc0pv0)

[**6. SUPUESTOS Y ACLARACIONES DE DISEÑO 8**](#_ess38n1b0gfr)

[**7. SCRIPTS ENTREGABLES 9**](#_4vs39kt1lvbv)

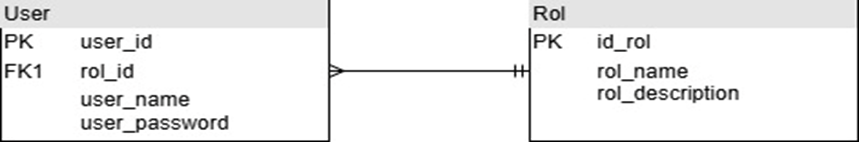
[**8. RELACIÓN CON EL MODELO DE CLASES 12**](#_r6xwvo861gmd)

# 

# 1. INTRODUCCIÓN

Este documento describe el diseño de la base de datos implementado para el sistema de gestión de usuarios del Proyecto ABP. El objetivo principal es garantizar una gestión eficiente de usuarios y el control de acceso mediante roles (admin/user) y garantizar integridad y seguridad (contraseñas hasheadas). La documentación incluye el Diagrama Entidad-Relación (DER), el modelo relacional, descripciones técnicas de cada entidad, y las justificaciones de diseño que sustentan las decisiones tomadas frente a ambigüedades. Aunque en la Evidencia 3 (EV3) la aplicación de consola no se conectará directamente a esta base, el diseño se ha preparado pensando en una futura integración con sistemas persistentes.

# 2. DIAGRAMA ENTIDAD-RELACIÓN (DER)

****

Este diagrama representa las entidades User y Rol, junto con la relación entre ellas.

* *User* (Usuario): Representa a los usuarios del sistema.
* *Rol:* Define los roles que pueden tener los usuarios.

**Relaciones:**

* La entidad *User* tiene una relación de muchos a uno con la entidad *Rol*.
* Esto implica que cada usuario posee un único rol, mientras que un rol puede estar asignado a múltiples usuarios.

# 

# 3. MODELO RELACIONAL

## **Tabla: Rol**

Esta tabla almacena los diferentes roles que pueden tener los usuarios en el sistema. Cada rol define un nivel de permisos y acceso.

| **ATRIBUTO** | **TIPO** | **CLAVE** | **RESTRICCIONES** | **DESCRIPCIÓN** |
| --- | --- | --- | --- | --- |
| ***rol\_id*** | **INT** | **PK** | **AUTO\_INCREMENT, NOT NULL** | **Valor autogenerado que identifica de forma única cada rol** |
| ***rol\_name*** | **VARCHAR(50)** | **UK** | **NOT NULL, UNIQUE** | **Nombre del rol ( admin, user)** |
| ***rol\_description*** | **VARCHAR(255)** | **-** | **NULL** | **Texto explicativo sobre permisos y alcance (opcional)** |

**Aclaraciones:**

1. **Roles fijos vs. dinámicos:**

Se optó por roles predefinidos ('Administrador', 'Usuario Estándar') sin capacidad de creación dinámica, basado en:

* Simplicidad del sistema actual
* No se especificó necesidad de gestión de roles en los requisitos
* ***Alternativa considerada:*** Tabla de permisos independiente con relación muchos-a-muchos a roles (descartada por complejidad innecesaria)

1. **Unicidad de *rol\_name*:**

* Se impone UNIQUE para evitar duplicados que podrían causar ambigüedades en la autorización.

## **Tabla: User**

Contiene la información de los usuarios del sistema, incluyendo sus credenciales de acceso y el rol asignado.

| **ATRIBUTO** | **TIPO** | **CLAVE** | **RESTRICCIONES** | **DESCRIPCIÓN** |
| --- | --- | --- | --- | --- |
| ***user\_id*** | INT | PK | AUTO\_INCREMENT, NOT NULL | Identificador numérico único para cada usuario |
| ***rol\_id*** | INT | FK | NOT NULL | Rol.rol\_id. Rol asignado al usuario (relación muchos-a-uno) |
| ***user\_name*** | VARCHAR(100) | UK | NOT NULL, UNIQUE | Nombre de usuario para autenticación (único) |
| ***user\_password*** | VARCHAR(255) | - | NOT NULL | Almacena hash de contraseña (no texto plano) |

**Aclaraciones:**

**1. Relación User-Rol:**

Se implementó como **muchos-a-uno** (no muchos-a-muchos) porque:

* Los requisitos no mencionan necesidad de múltiples roles por usuario
* Simplifica la consulta de permisos (un solo rol determina todos los accesos)
* ***Proyección***: Si futuras iteraciones requieren roles múltiples, será necesario:
  + Crear tabla intermedia user\_roles
  + Modificar lógica de autorización

**2. Seguridad de contraseñas:**

* Aunque el campo es VARCHAR(255), se asume que:
* La aplicación implementará hash (ej: bcrypt) antes de almacenar
* Se reservan 255 caracteres para compatibilidad con algoritmos de hash modernos
* ***Justificación:*** No se usa tipo "password" específico por portabilidad entre motores de BD

**3. Unicidad de user\_name:**

* Garantiza que no existan duplicados que compliquen el proceso de autenticación.

# 4. DESCRIPCIÓN DE ENTIDADES

**Rol**

**Propósito:** Define los diferentes tipos de usuarios que pueden acceder al sistema, diferenciando sus niveles de permisos.

**Contenido:**

- *rol\_name* : nombres como 'Administrador' o 'Usuario Estándar'.

- *rol\_description*: texto explicativo del alcance de ese rol.

**Justificación:**

* Se asume un conjunto finito y definido de roles, por lo que no se previó la posibilidad de creación dinámica de roles por parte de usuarios. Se modeló como una entidad separada por normalización.

**User**

**Propósito:** Representa a los usuarios del sistema.

**Contenido:**

- *user\_name* : único por usuario, utilizado para autenticación.

- *user\_password* : valor encriptado (hash) para seguridad.

- *rol\_id* : referencia al tipo de usuario.

**Justificación:**

-Se decidió modelar una relación uno a muchos entre Rol y User bajo la suposición de que cada usuario puede tener un solo rol activo en el sistema.

-Aunque en sistemas más complejos podría justificarse una relación muchos a muchos entre usuarios y roles, dicha complejidad no es necesaria aquí.

-La autenticación y autorización estarán controladas por este vínculo directo y único.

# 5. NORMALIZACIÓN DE LA BASE DE DATOS (HASTA LA TERCERA FORMA NORMAL - 3NF)

EL diseño actual de la base de datos **ProyectoABP\_2025** ha sido analizado y cumple con los requisitos de la Tercera Forma Normal (3NF), un estándar común y eficiente para la mayoría de las aplicaciones.

**5.1. Formas Normales Aplicadas:**

* **Primera Forma Normal (1NF):**
  + **Requisito:** Cada columna debe contener valores atómicos (indivisibles), y cada fila debe ser única (poseer una clave primaria).
  + **ProyectoABP\_2025:**
    - **Tabla Rol:** Las columnas **rol\_id, rol\_name,** y **rol\_description** contienen valores atómicos. **rol\_id** actúa como clave primaria, asegurando la unicidad de cada registro de rol.
    - **Tabla User:** Las columnas **user\_id, rol\_id, user\_name,** y **user\_password** contienen valores atómicos. **user\_id** actúa como clave primaria, garantizando la unicidad de cada registro de usuario
* **Segunda Forma Normal (2NF):**
  + **Requisito:** Debe estar en 1NF, y todas las columnas no clave deben depender completamente de la clave primaria completa.
  + **ProyectoABP\_2025:**
    - Ni la tabla **Rol** ni la tabla **User** utilizan claves primarias compuestas; sus claves primarias (**rol\_id y user\_id** respectivamente) son simples. Por lo tanto, no hay dependencias parciales de claves primarias compuestas.
* **Tercera Forma Normal (3NF):**
  + **Requisito:** Debe estar en 2NF, y no debe haber dependencias transitivas de las columnas no clave. Es decir, ningún atributo no clave debe depender de otro atributo no clave; todos deben depender directamente de la clave primaria.
  + **ProyectoABP\_2025:**
    - **Tabla Rol:** **rol\_name** y **rol\_description** (atributos no clave) dependen directamente de **rol\_id** (la clave primaria). No hay atributos no clave que dependan de otros atributos no clave.
    - **Tabla User:** **rol\_id**, **user\_name**, y **user\_password** (atributos no clave) dependen directamente de **user\_id** (la clave primaria). **rol\_id** es una clave foránea que establece una relación con la tabla **Rol**.

**5.2. Beneficios de la Normalización a 3NF en este Diseño:**

* **Reducción de Redundancia:** La información de los roles se almacena una sola vez en la tabla **Rol**, y la tabla **User** solo hace referencia a ella mediante **rol\_id**, evitando repetir nombres o descripciones de roles para cada usuario.
* **Mejora de la Integridad:** Asegura que los cambios en la descripción de un rol se realicen en un único lugar (tabla **Rol**), y que todos los usuarios con ese rol reflejen automáticamente la actualización. Además, previene la inserción de usuarios con roles inexistentes.
* **Minimización de Anomalías:** Reduce las posibilidades de anomalías de inserción, actualización y eliminación de datos.

# 6. SUPUESTOS Y ACLARACIONES DE DISEÑO

**6.1. Cardinalidad fija entre *User* y *Rol*:**

* **Supuesto:** Cada usuario tendrá un único rol. No se requiere una relación muchos a muchos ni una tabla intermedia.
* **Justificación:** Esta decisión se basa en la simplicidad del sistema descrito y el análisis de los requerimientos funcionales provistos en la lógica de programación.

**6.2. Contraseñas:**

* Aunque en el modelo se define el campo *user\_password* como VARCHAR(255), se asume que en la aplicación real este campo almacenará contraseñas hasheadas, y no texto plano.
* La gestión de contraseñas seguras deberá implementarse a nivel de backend o capa de lógica de aplicación.
* **Justificación:** El alcance del proyecto no requiere personalización avanzada de roles.

**6.3. Roles iniciales predefinidos:**

* Se insertan de forma predeterminada los roles 'Administrador' y 'Usuario Estándar'. El sistema no contempla, por ahora, una interfaz de gestión dinámica de roles.
* **Justificación:** El alcance del proyecto no requiere personalización avanzada de roles.

# 7. SCRIPTS ENTREGABLES

Los siguientes archivos .sql acompañan esta documentación:

**01\_creacion\_base.sql**

* Creación de la base de datos ProyectoABP
* Creación de tablas Rol y User
* Inserción inicial de roles

SQL:

-- Script de Creación de la Base de Datos para la app del proyecto ABP de Introducción a la Programación y Base de Datos

-- 1. Crear la Base de Datos

**CREATE DATABASE IF NOT EXISTS** `ProyectoABP\_2025`;

-- 2. Seleccionar la Base de Datos

**USE** `ProyectoABP\_2025`;

-- 3. Crear la Tabla 'Rol'

-- Contiene los distintos tipos de roles de usuario

**CREATE TABLE** `Rol` (

`**rol\_id**` INT PRIMARY KEY AUTO\_INCREMENT,

`**rol\_name**` VARCHAR(50) NOT NULL UNIQUE,

`**rol\_description**` VARCHAR(255)

);

-- 4. Insertar roles iniciales

-- Son los roles predefinidos que los usuarios pueden tener

**INSERT INTO** `Rol` (`rol\_name`, `rol\_description`) **VALUES**

('admin', 'Puede visualizar, modificar y eliminar usuarios del sistema.'),

('user', 'Puede acceder y gestionar solo sus propios datos personales.');

-- 5. Crear la Tabla 'User'

-- Almacena la información de los usuarios del sistema

**CREATE TABLE** `User` (

`**user\_id**` INT PRIMARY KEY AUTO\_INCREMENT,

`**rol\_id**` INT NOT NULL,

`**user\_name**` VARCHAR(100) NOT NULL UNIQUE,

`**user\_password**` VARCHAR(255) NOT NULL,

**FOREIGN KEY** (`rol\_id`) **REFERENCES** `Rol`(`rol\_id`)

);

**02\_crud\_usuario.sql**

Este script contiene las operaciones básicas de Manipulación de Datos (CRUD: Create, Read, Update, Delete) sobre la tabla **User.**

* Inserciones de usuarios
* Lecturas: consulta por ID, por nombre, y con relación a roles
* Actualizaciones: modificación de contraseña y cambio de rol
* Eliminaciones por nombre de usuario e ID

**SQL**

-- Script de Consultas CRUD para la tabla 'User'

-- Seleccionar la Base de Datos

**USE** `ProyectoABP\_2025`;

-- Sección de Limpieza (Opcional, para re-ejecutar el script)

-- Elimina los usuarios de prueba si ya existen, para evitar errores de duplicidad.

**DELETE FROM** User **WHERE** user\_name **IN** ('admin\_test', 'user\_test');

-- Sección de Inserción (CREATE)

-- Crea nuevos registros de usuario en la tabla 'User'.

**INSERT INTO** User (rol\_id, user\_name, user\_password) **VALUES**

((**SELECT** rol\_id **FROM** Rol **WHERE** rol\_name ='admin'), 'admin\_test','hash\_admin\_123'),

((**SELECT** rol\_id **FROM** Rol **WHERE** rol\_name ='user'), 'user\_test','hash\_user\_abc');

-- Sección de Lectura (READ)

-- Recupera información de usuarios de la base de datos.

-- Leer todos los usuarios con su rol asociado

**SELECT** U.user\_id, U.user\_name, R.rol\_name

**FROM** User U

**INNER JOIN** Rol R **ON** U.rol\_id = R.rol\_id;

-- Leer un usuario específico por nombre de usuario

**SELECT** user\_id, user\_name, rol\_id

**FROM** User

**WHERE** user\_name = 'admin\_test';

-- Leer un usuario por su ID

**SELECT** user\_name, rol\_id

**FROM** User

**WHERE** user\_id = 1;

-- Sección de Actualización (UPDATE)

-- Modifica datos existentes de usuarios en la tabla 'User'.

-- Cambiar la contraseña de un usuario (ej. 'user\_test')

**UPDATE** User

**SET** user\_password = 'nuevo\_hash\_de\_user\_test'

**WHERE** user\_name = 'user\_test';

-- Cambiar el rol de un usuario (ej. 'user\_test' a 'admin')

**UPDATE** User

**SET** rol\_id = (**SELECT** rol\_id **FROM** Rol **WHERE** rol\_name = 'admin')

**WHERE** user\_name = 'user\_test';

-- Sección de Eliminación (DELETE)

-- Elimina registros de usuario de la tabla 'User'.

-- Eliminar un usuario por nombre de usuario

**DELETE FROM** User

**WHERE** user\_name = 'user\_test';

-- Eliminar un usuario por ID

**DELETE FROM** User **WHERE** user\_id = 2;

# 8. RELACIÓN CON EL MODELO DE CLASES

El diseño de esta base de datos se apoya en el modelo de clases provisto en la documentación de introducción a la programación. Particularmente:

* La clase *User* define los atributos principales que se reflejan en la tabla homónima.
* La relación entre *Admin* y *User* se traduce en el uso del campo *rol\_id*, no mediante herencia directa, por las limitaciones del modelo relacional.
* Las funciones como *register\_user*, *login*, *change\_role* y *delete\_user* se corresponden con las operaciones de inserción, lectura, actualización y eliminación implementadas en los scripts SQL.