TECNICATURA UNIVERSITARIA
EN PROGRAMACIÓN
A DISTANCIA

PROGRAMACIÓN 2

TRABAJO PRÁCTICO INTEGRADOR

GRUPO 18

Integrantes: **Hugo Catalan, Ignacio Carné, Matias Carro, Gabriel Carbajal**

Materia: Programación 2
Comision 11
Grupo 18

Link al repositorio del trabajo:
https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18
Link al video del trabajo:
https://www.youtube.com/watch?v=JzoYeIVNUH0

Trabajo Práctico Integrador Programación 2

Elección del dominio y justificación

El dominio elegido para este proyecto es la gestión de usuarios y credenciales, un componente esencial en cualquier sistema informático que requiera autenticación, autorización y control de acceso. Este enfoque permite modelar entidades como usuarios, credenciales de acceso, estados de cuenta y reglas de seguridad, aplicando principios de diseño orientado a objetos y arquitectura modular.

La elección se justifica por su relevancia práctica: este tipo de lógica se encuentra en aplicaciones empresariales, sistemas administrativos, plataformas web y entornos corporativos.
Además, permite abordar temas clave como:

- Validación de datos y reglas de negocio (campos obligatorios, unicidad de username/email)
- Seguridad en el manejo de contraseñas (hash + salt)
- Implementación de transacciones para garantizar consistencia entre entidades relacionadas
- Aplicación de patrones DAO y Service para desacoplar lógica y persistencia

Desde una perspectiva formativa, este proyecto permite asentar las bases de nuestro futuro profesional, consolidando habilidades en Java, JDBC, MySQL, UML y diseño de capas.

Integrantes y Roles

1. Gabriel Carbajal – Base de datos y Menú principal

- Diseño y documentación de las tablas (usuario, credencial_acceso).
- Definición de claves primarias, foráneas y restricciones (PK, FK, UNIQUE, CHECK).
- Preparación de scripts SQL iniciales y reproducibles.
- Diseño del AppMenu y Main para iniciar la aplicación.
- Diagramas, apoyo en UML.

2. Ignacio Carné – DAO y UML

- Implementación de interfaces UsuarioDao y CredencialAccesoDao.
- Programación de consultas CRUD con JDBC y PreparedStatement.
- Manejo de excepciones de acceso a datos (DataAccessException).
- Validación de operaciones verificando que las operaciones respeten las condiciones del sistema.
- Elaboración de diagramas UML actualizados y consistentes con el código.

3. Matías Carro – Servicios y Flujo de uso

- Implementación de UsuarioServiceImpl y CredencialAccesoServiceImpl.
- Manejo de transacciones (commit y rollback).
- Aplicación de validaciones (unicidad de username/email, estado ACTIVO/INACTIVO).
- Creación de usuario más credencial en una operación atómica.
- Documentación del flujo de uso y pruebas de la aplicación.
- Apoyo en DAO para integración con Services.

4. Hugo Catalan – Integración y Utilitarios

- Integración de los servicios en el menú (alta, baja, modificación, consulta).
- Implementación de PasswordUtil (hash + salt).
- Manejo de rollback en operaciones de prueba/demostración.
- Validación de login y actualización de credenciales.
- Apoyo en pruebas de integración y casos de uso.

Creación del documento, README y javadoc realizado por todo el grupo

Arquitectura por capas

1. Capa de presentación (App/UI)

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/app

Responsabilidad:

- Interacción con el usuario (menú, entrada de datos, mensajes)
- Coordinación de acciones entre servicios y entidades

2. **Capa de configuración**

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/config

Responsabilidad:

- Manejo de parámetros globales y conexión a la base de datos

3. **Capa de entidades (modelo de dominio)**

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/entities

Responsabilidad:

- Representación de las tablas como clases Java
- Encapsulan atributos y relaciones del modelo

4. **Capa de acceso a datos (DAO)**

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/dao
https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/dao/impl

Responsabilidad:

- Operaciones CRUD sobre la base de datos
- Implementación de consultas SQL

    - Interfaces: UsuarioDao.java, CredencialAccesoDao.java, GenericDao.java
    - Implementaciones: UsuarioDaoImpl.java, CredencialAccesoDaoImpl.java

5. **Capa de servicios (lógica de negocio)**

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/servicehttps://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/service/impl

Responsabilidad:

- Validaciones de negocio
- Orquestación de DAOs
- Manejo de transacciones

    - Interfaces: UsuarioService.java, CredencialAccesoService.java, GenericService.java
    - Implementaciones: UsuarioServiceImpl.java, CredencialAccesoServiceImpl.java

6. **Capa de utilitarios**

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/util

Responsabilidad:

- Funciones auxiliares como generación de salt y hashing de contraseñas
  Utilitario: PasswordUtil.java

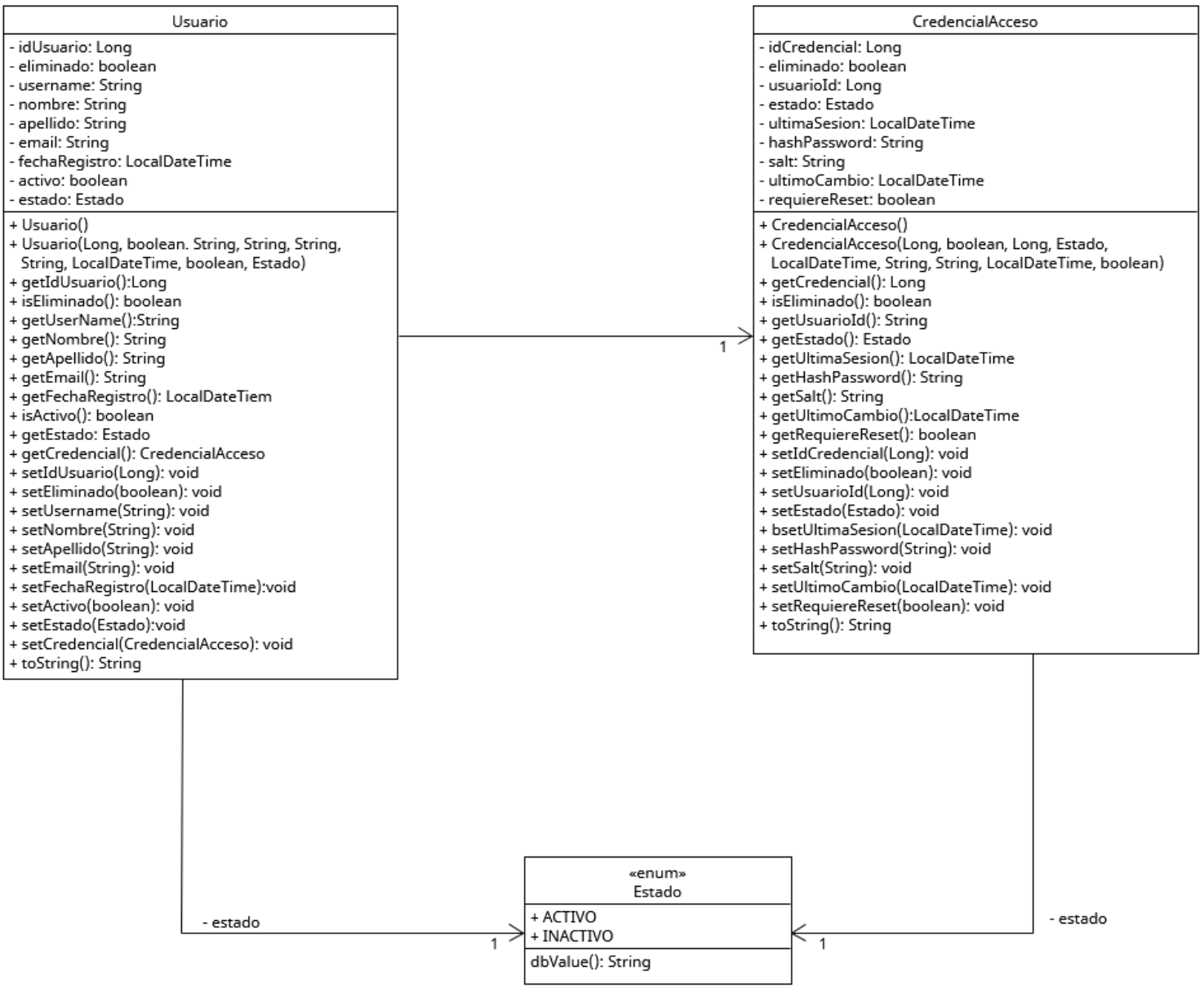
7. **Capa de excepciones**

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/tree/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/exceptions

Responsabilidad:

- Manejo de errores personalizados en acceso a datos

Diagrama UML:



Relación 1 → 1 (Usuario → CredencialAcceso): Se implenta la relación como unidireccional desde Usuario hacia CredencialAcceso.

- **En Java:** Usuario tiene un atributo credencial, mientras que CredencialAcceso solo guarda el usuarioId como FK.

- **En Base de Datos:** la tabla credencial_acceso tiene usuario_id con restricción UNIQUE, asegurando que cada usuario tenga una sola credencial.

FK única vs PK compartida:

FK única (usuario_id UNIQUE) en credencial_acceso. Esto simplifica la gestión de claves, mantiene independencia entre tablas y asegurando que cada usuario tenga una sola credencial única (y viceversa)

Estructura de la Base de Datos

El sistema se apoya en una base de datos MySQL 8.0 con las siguientes entidades principales:

- **Usuario**
  - id_usuario (PK, autoincremental)
  - username, nombre, apellido, email (con restricciones de unicidad en username y email)
  - fecha_registro, activo, estado (con CHECK en estado: ACTIVO/INACTIVO)
  - eliminado (para baja lógica)
- **CredencialAcceso**
  - id_credencial (PK, autoincremental)
  - usuario_id (FK, UNIQUE, asegura relación 1:1 con Usuario)
  - hash_password, salt, ultimo_cambio, requiere_reset
  - estado (ACTIVO/INACTIVO, con CHECK)
  - eliminado (para baja lógica)
- **Estado (catálogo)**
  - id_estado (PK)
  - nombre_estado (ACTIVO/INACTIVO)
  -

Integridad referencial:

- credencial_acceso.usuario_id es la FK hacia usuario.id_usuario con ON DELETE CASCADE.
- Restricción UNIQUE en usuario_id asegura que cada usuario tenga como máximo una credencial.

Triggers y procedimientos:

- Triggers sincronizan estado entre usuario y credencial.
- Stored procedure sp_actualizar_password_seguro actualiza hash y salt de forma segura.

Orden de Operaciones

- **Crear Usuario Simple:** Inserta en usuario con valores iniciales (activo, eliminado=false, estado). Se recupera el id_usuario generado con RETURN_GENERATED_KEYS.
- **Crear Credencial para Usuario Existente:** Inserta en credencial_acceso con FK hacia usuario_id. Se calcula hash+salt antes de persistir.
- **Crear Usuario + Credencial (Transacción):**
    - Inserta usuario.
    - Recupera id_usuario.
    - Inserta credencial asociada con ese ID.
    - Commit al final si ambas operaciones son exitosas.
    - Rollback si ocurre error en cualquiera de los pasos.
- **Actualizar Usuario/Credencial:** Operaciones con UPDATE sobre campos específicos. Se valida estado y unicidad antes de persistir.
- **Eliminar Usuario:**
    - Baja lógica: UPDATE usuario SET eliminado=TRUE.
      "Elimina" el usuario del programa, lo "esconde" pero aun persiste en la Base de DAtos.
    - Baja física: DELETE FROM usuario WHERE id_usuario = "..." esto elimina también la credencial asociada por ON DELETE CASCADE.
      Elimina el usuario, junto a la credencial que le corresponde.

Transacciones (Commit/Rollback)

El sistema implementa transacciones en operaciones críticas para garantizar la consistencia de los datos. Un conjunto de operaciones se ejecuta como una unidad: o se completan todas con éxito (commit) o ninguna se aplica (rollback).

Proceso de transacción, se utiliza **createUsuarioConCredencial** Dentro del paquete **UsuarioServiceImpl**

https://github.com/MatiasManuelCarro/UTN-TUPaD-P2_TPI_Grupo18/blob/main/TPI/programacion2/src/main/java/integradorfinal/programacion2/service/impl/UsuarioServiceImpl.java

Este método crea un **usuario** y su **credencial de acceso** en una sola transacción:

- Se desactiva el autocommit, esto evita que cada sentencia SQL se confirme automáticamente.
- Se inserta primero el usuario en la tabla usuario y luego se inserta la credencial asociada en la tabla credencial_acceso, usando el id_usuario como FK.
- Si ambas operaciones son exitosas, se confirma la transacción y los cambios quedan permanentes.

Si ocurre cualquier error en el proceso, se revierte la transacción completa, evitando que quede un usuario sin credencial o una credencial huérfana.

Reglas de negocio - Gestión de Usuarios y Credenciales (Base de datos)

Identificación única de usuarios

- Cada usuario debe tener un usuario único en el sistema.
- Cada usuario debe registrar un email único

.

Estado del usuario

- El campo activo solo puede tomar los valores: ACTIVO, INACTIVO

Relación usuario–credencial

- Toda credencial debe estar asociada a un usuario existente (usuario_id obligatorio).
- Un usuario puede tener una sola credencial activa a la vez.

Estado de credenciales

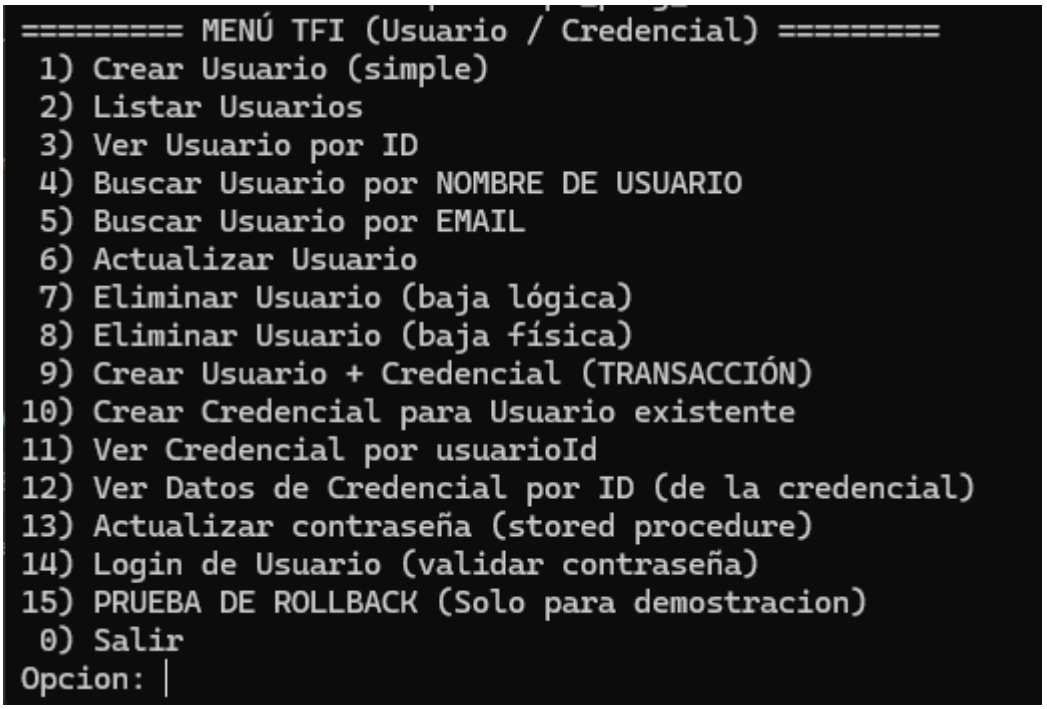
- El campo estado de la credencial solo puede ser ACTIVO, INACTIVO

Seguridad e integridad de datos

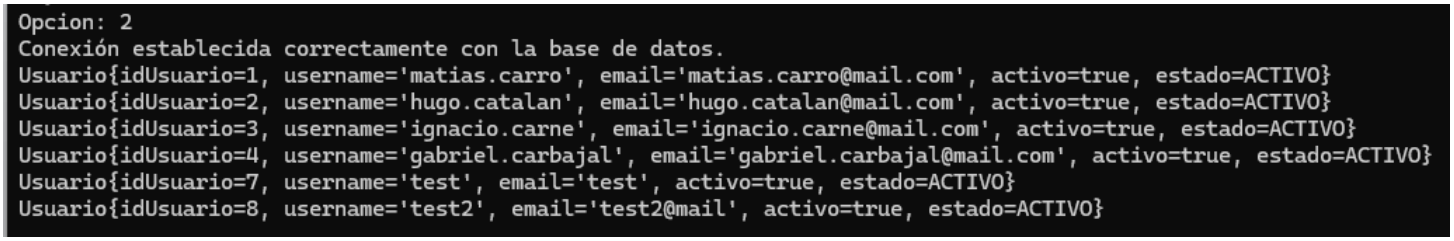
- La eliminación de un usuario implica la eliminación automática de sus credenciales asociadas (ON DELETE CASCADE).
- Las actualizaciones de id_usuario se propagan a las credenciales (ON UPDATE CASCADE).

Pruebas:

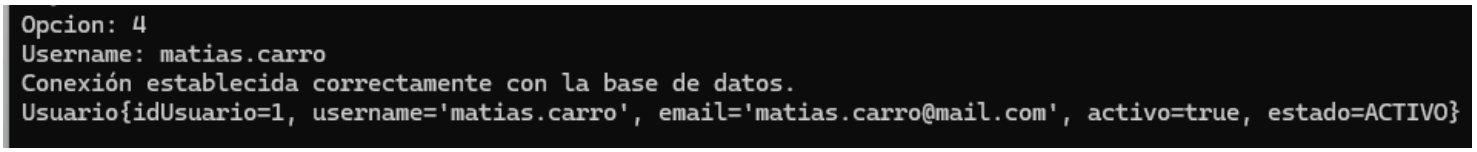
Capturas del menú:



Consultas SQL:
Listar Usuario

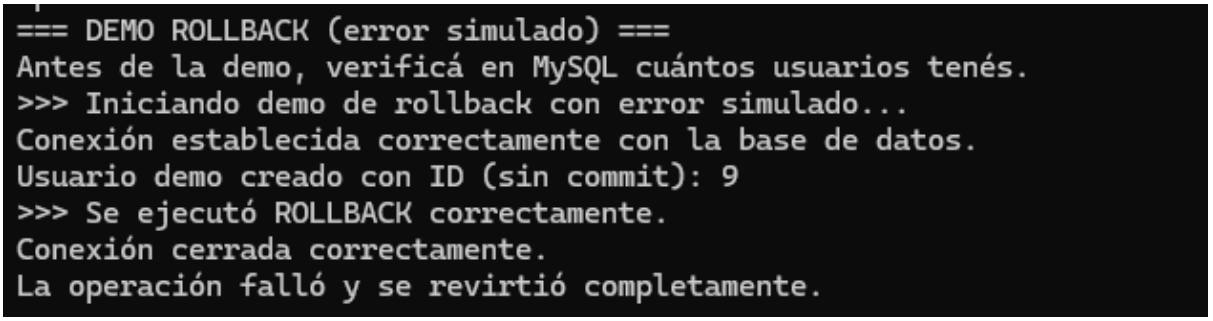


Buscar por nombre de usuario:



Buscar por ID:

Rollback:



Se simula un rollback en:
https://github.com/Hugocatalan/UTN-TUPaD-P2_TPI_Grupo18/blob/6d47efefd5cd2dddb4c1d037fea364cd43ee188d/TPI/programacion2/src/main/java/integradorfinal/programacion2/service/impl/UsuarioServiceImpl.java#L245

Durante la transacción simulada se ejecuta:

```
throw new SQLException("Error simulado para demostrar ROLLBACK");
```

para simular un error, de esta forma mostrando cómo se maneja un rollback en el sistema.

Conclusiones

- El proyecto logró implementar una arquitectura organizada con capas DAO, Service y AppMenu, asegurando separación de responsabilidades.
- Se garantizó la relación 1→1 entre Usuario y Credencial.
- Las operaciones CRUD funcionan correctamente, con soporte para transacciones (commit/rollback) que mantienen la consistencia de los datos.
- Se aplicaron buenas prácticas de seguridad: contraseñas con hash y salt, procedimientos almacenados y control de estados.
- La documentación (README, UML, informe) y la presentación en video complementan el desarrollo técnico.

Mejoras futuras

- **Validaciones más robustas:** agregar restricciones adicionales (ej. longitud mínima de contraseña, formato de email).
- **Interfaz gráfica o web:** Posibilidad futura de agregar una interfaz.
- **Devolver información con mejor formato.** Retornar la info al usuario de manera más textual y comprensible.

Fuentes:

- Obregón, A. (s. f.). Creating user menus in Java with loops and switch. Medium. https://medium.com/@AlexanderObregon/creating-user-menus-in-java-with-loops-and-switch-040149bd9732

- Oracle. (s. f.). Using transactions (The Java Tutorials > JDBC Database Access > JDBC Basics). Oracle Docs. https://docs.oracle.com/javase/tutorial/jdbc/basics/transactions.html

- GeeksforGeeks. (s. f.). Java.util package in Java. GeeksforGeeks. https://www.geeksforgeeks.org/java/java-util-package-java/

- Apache Software Foundation. (s.f.). *Maven in 5 minutes*. Maven Project. Recuperado el 16 de noviembre de 2025, de https://maven.apache.org/guides/getting-started/maven-in-five-minutes.html

Herramientas utilizadas

- **IDE y Lenguaje:** NetBeans / IntelliJ IDEA con Java 17.
- **Gestión de dependencias:** Maven.
- **Base de datos:** MySQL 8.0.
- **Diagramación:** UML y DER (draw.io / StarUML).
- **Control de versiones:** GitHub (repositorio público).
- **Documentación:** README.md, Javadoc, informe técnico.
- **Asistencia con IA:** Microsoft Copilot, utilizado como asistencia para reformular explicaciones, organizar documentación y apoyo en conceptos técnicos.