UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Final Integrador (TFI) – Programación II

Alumnos

Gonzalo Prados - gonza.prados@gmail.com

Maximiliano Niemiec - maximiliano.niemiec@tupad.utn.edu.ar

Nicolas Viruel – nicolas.viruel@tupad.utn.edu.ar

Paola Pasallo - paola.pasallo@tupad.utn.edu.ar

Tecnicatura Universitaria en Programación - Universidad Tecnológica Nacional.

17 de noviembre de 2025

Rol por integrante

Efectuamos una división de tareas. Cada sección del proyecto fue revisada en grupo antes de efectuar los commits al repositorio original.

Nicolás Viruel fue el encargado de la elaboración del diagrama UML de clases, definición de los atributos, métodos, visibilidades y la relación 1 → 1 entre las entidades de dominio. Creación y configuración del proyecto Gradle, implementación de la conexión a la base de datos y la creación de esta.

Paola Pasallo fue la encargada del desarrollo del modelo de datos del sistema, creando las clases centrales (Base, Legajo, Empleado y EstadoLegajo), y desarrollo de los componentes correspondientes a la capa DAO (Generic, Legajo, Empleado).

Gonzalo Prados se encargó de desarrollar los componentes correspondientes a la capa de servicios, capa intermediaria entre los DAOS y el menú de la aplicación, concentrando la lógica de negocio, las validaciones y la gestión de transacciones, desarrollando los siguientes archivos: GenericService, LegajoService y EmpleadoService.

Maximiliano Niemiec estuvo a cargo del desarrollo de la capa Main (Presentación / Consola). Creación de la interfaz de interacción del usuario con el sistema a través de la consola. Desarrollo de Main, AppMenu, MenuHandler y MenuDisplay.

Todos los integrantes efectuaron pruebas del sistema por separado para corroborar su funcionamiento.

Elección del dominio

Como elección de dominio tomamos el par Empleado → Legajo en tanto que presenta similitudes con los escenarios ficticios que planteamos en el marco del TPI de Bases de Datos. Continuando con ese escenario ficticio planteamos el desarrollo de una aplicación para control de personal dentro de una empresa.

La aplicación permite crear nuevos empleados, listarlos, actualizar información, eliminarlos (darlos de baja), creación de legajos y listarlos.

Diseño: decisiones clave

Se ha modelado una relación 1:1 unidireccional entre las entidades empleado y legajo, utilizando clave foránea única en lugar de clave primaria compartida. La decisión de claves primarias separadas y FK opcional permite considerar escenarios tales como empleados en proceso de contratación sin afectar la integridad del modelo. Este patrón provee una separación clara entre datos de identidad personal y la información administrativa laboral.

Estructura de la relación

Tabla EMPLEADO

PK independiente: id BIGINT AUTO_INCREMENT

FK única: legajo_id BIGINT UNIQUE (puede ser NULL)

Tabla LEGAJO

PK independiente: id BIGINT AUTO_INCREMENT

Datos laborales: nro_legajo, categoría, estado, fecha_alta, observaciones

Sin referencia inversa: Diseño unidireccional

Arquitectura por capas

Cada capa tiene una responsabilidad única y los cambios en una capa no afectan a la otra. Cada capa se puede testear de forma independiente y es relativamente simple agregar funcionalidades nuevas.

Por un lado, la capa Models/ (Dominio) es el encargado de representar entidades del negocio. Contienen sólo datos y validaciones básicas. En nuestro caso particular contiene las Clases: Empleado, Legajo, EstadoLegajo y Base.

La capa Dao/ (Persistencia), es la responsable del acceso a la base de datos y las operaciones CRUD con JDBC para la conexión con MySQL. También, aisla la lógica de la base de datos. En nuestra estructura contiene: EmpleadoDAO, LegajoDAO y GenericDAO.

Dentro de la capa Service/(Lógica de Negocio), se realizan múltiples funciones: Coordinación de múltiples DAOs, manejo de transacciones (commit/rollback), aplicación de reglas de negocio y orquesta operaciones complejas. Nuestra aplicación contiene: EmpleadoService, LegajoService y GenericService.

Por último, la capa Main/(Presentación) también tiene muchas funciones: capturar entradas del usuario, mostrar resultados y menús, delega la lógica a los Servicios y gestiona el flujo de la aplicación.

Persistencia: estructura de la base de datos.

Tenemos las tablas empleado y legajo, relacionadas 1 a 1 mediante legajo_id.

TABLA empleado (id, nombre, apellido, dni, legajo_id)

↓

TABLA legajo (id, nro_legajo, estado, categoría)

Orden de las operaciones:

En la creación primero empleado y luego legajo (en caso de ser necesario).

Para la eliminación usamos un borrado lógico.

PARA CREAR:

1. Primero → Creo el empleado

2. Después → Crear el legajo. (Un empleado puede no tener asignado un legajo)

PARA ELIMINAR:

1. Primero → ingresar el ID del empleado a eliminar

2. Después → Se elimina el empleado de la lista

Transacciones

El commit/rollback se realiza en la capa Service, específicamente en métodos como crear() o actualizar().

Validaciones y reglas de negocio

La implementación del proyecto sigue un diseño de validaciones distribuidas en tres capas, asegurando la integridad de los datos y el cumplimiento de las reglas de negocio.

1 - Base de datos

Utilizamos restricciones básicas para asegurar la consistencia de los datos almacenados

NOT NULL en campos como DNI y NOMBRE en la tabla empleado

UNIQUE en DNI de la tabla empleado o nro_legajo de la tabla legajo para garantizar la unicidad de dichos datos

Soft Delete: Campo “eliminado” para eliminación lógica

2 - Capa DAO

Es la capa responsable de la interacción con SQL. Algunas acciones de validación y gestión que podemos encontrar en ella son:

Manejo de valores nulos y conversión segura de tipos de datos (Ej. fechas locales con LocalDate)

Liberación de recursos: asegura el cierre de la conexión a base de datos luego de utilizar por ejemplo la sentencia “PreparedStatement” usando “try-with-resources”.

3 - Nivel de Servicios

En la capa que actúa como reguladora de las reglas de negocio, coordinando transacciones y lanzando excepciones se utilizan:

Validación de Integridad de Datos: Se verifica que los objetos de dominio (ej., Empleado) no sean nulos y que contengan valores para todos los campos obligatorios del negocio.

Commit/Rollback: Utiliza excepciones que nos aseguran en caso de que una operación falle (Ej: crear un empleado con DNI duplicado), otra operación anterior (Ej: haber creado un legajo) se deshaga.

Pruebas realizadas (capturas del menú y consultas SQL útiles)

- Transacción: Utilizando el menú principal interactivo con el usuario crearemos un empleado y un legajo

```
========= MENU PRINCIPAL =========
1. Crear Empleado
2. Listar Empleados
3. Actualizar Empleado
4. Eliminar Empleado (baja lógica)
5. Crear Legajo
6. Listar Legajos
0. Salir
Seleccione una opción:
```

```
--- Crear Empleado ---
Nombre: Jose
Apellido: Belgrano
DNI: 38985326
Email: jose_b@gmail.com
Área: Sistemas

--- Datos del Legajo ---
Nro Legajo: 389853268
Categoría: jr
 Empleado y Legajo creados correctamente!
```

Relación 1:1: Veremos si está en nuestra lista

```
Seleccione una opción: 2

--- Listado de Empleados ---
Empleado{id=5, nombre='Jose', apellido='Belgrano', dni='38985326', email='jose_b@gmail.com', fechaIngreso=2025-11-15, area='Sistemas', legajo=389853268}
```

Chequeamos con MySQL si está en nuestra Base de datos

```
SELECT e.id, e.nombre, e.apellido, l.nro_legajo, l.estado
FROM empleado e
JOIN legajo l ON e.legajo_id = l.id;
```

	id	nombre	apellido	nro_legajo	estado
▶	5	Jose	Belgrano	389853268	ACTIVO

Rollback: intentaremos crear un legajo ya existente para forzar un error

```
--- Crear Legajo ---
Nro Legajo: 38985326
Categoría: jr
```

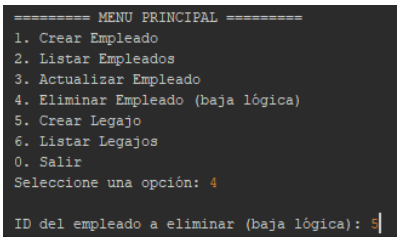
```
Error: Error al crear empleado: Duplicate entry '38985326' for key 'legajo.nro_legajo'
```

id	eliminado	nro_legajo	categoria	estado	fecha_alta
8	0	389853268	jr	ACTIVO	2025-11-15
6	0	38985326	jr	ACTIVO	2025-11-15

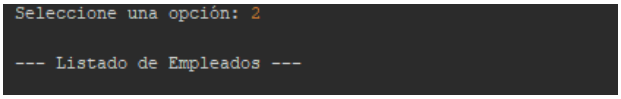
Vemos que no hay dos legajos iguales. No se agregó por duplicado ya que la restricción definida en N° legajo funcionó correctamente.

Baja lógica

Daremos de baja el usuario creado. No se borrará de nuestra base de datos. Podríamos recuperar sus datos en un futuro si así lo deseamos.



En la siguiente imagen veremos que el empleado ya no está en la lista de empleados



Y Chequeando en MySQL vemos que el empleado sigue estando en la base de datos, pero en la columna "eliminado" nos figura el número 1 confirmando que existe la baja lógica.

	id	nombre	apellido	dni	eliminado
▶	3	maxi	maxi	1241412	1
	4	jajaja	asdasdas	4564564	1
	5	Jose	Belgrano	38985326	1

Además, en la imagen se ve otros empleados agregados a modo de prueba

Conclusiones

En este trabajo pudimos tener un primer acercamiento a un producto real de programa. Pudimos familiarizarnos con la estructura de capas y servicios, conceptos que hasta este momento nos eran completamente extraños. También, logramos realizar una integración con una base de datos.

A medida que avanzábamos en el trabajo e íbamos uniendo todo nos daba la sensación de que cada uno estaba trabajando en partes distintas de un mismo rompecabezas y una imagen se iba haciendo cada vez más clara.

También tuvimos un primer acercamiento a problemas de la cotidianeidad de un programador: problemas de compatibilidad, errores que no podíamos reproducir, diferencias de configuraciones en nuestros sistemas particulares que contribuían a nuestra confusión. Afortunadamente pudimos ayudarnos entre nosotros a poner a punto nuestros IDE, nuestras bases de datos y nuestras configuraciones particulares para llegar a buen puerto.

Estrictamente hablando sobre los contenidos del trabajo pudimos apreciar las ventajas y dificultades del trabajo en grupo en programación y la arquitectura en capas.

Reflexión sobre posibles mejoras

El sistema implementa un modelo inicial de gestión de empleados y legajos con una relación 1→1. Actualmente permite crear un empleado sin legajo, lo cual es útil en situaciones de preselección. Sin embargo, una vez creado el empleado, el sistema no permite asignarle un legajo de manera posterior, limitando la completitud de la relación. De manera similar, es posible crear un legajo sin vincularlo a un empleado, pero este registro no puede asociarse luego a un empleado nuevo ni existente, quedando aislado dentro del dominio.

Por otra parte, cuando se elimina lógicamente un empleado, el sistema no actualiza el estado del legajo asociado, el cual permanece activo, lo que puede generar inconsistencias entre el estado del empleado y el del registro administrativo.

A partir de estas observaciones, se identifican diversas líneas de mejora futura:

Validaciones de integridad: incorporar controles sobre formato de DNI, correo electrónico y longitudes máximas de los campos.

Reglas de dominio: validar fechas para evitar valores futuros o no válidos.

Validaciones de estado: evitar la eliminación de empleados que posean legajos activos.

Reglas de unicidad en la aplicación: agregar validaciones previas para detectar duplicados antes de ejecutar la operación en la base de datos.

Mejoras en la gestión de la relación empleado–legajo:

permitir la asignación posterior de un legajo a un empleado creado sin él, habilitar la vinculación de legajos independientes con empleados existentes, sincronizar el estado del legajo cuando el empleado es marcado como eliminado.

Estas mejoras fortalecerían la coherencia del modelo de datos, el cumplimiento de las reglas de negocio y la consistencia general del sistema en futuras versiones.

Link Repositorio Github

Link video explicativo

Fuentes consultadas:

Baeldung. (2022). *Implementing the DAO Pattern in Java*. https://www.baeldung.com/java-dao-pattern

GitBook. *DAO (Data Access Object).* https://mcazorla.gitbooks.io/programacion-en-el-servidor/content/patrones_de_diseno_en_php_i/dao_data_access_object.html

IBM. (2015). *Capa de Servicios.* https://www.ibm.com/docs/es/iis/11.5.0?topic=components-services-tier

TutorialsPoint. (2023). *CRUD Operations Using JDBC.* https://www.tutorialspoint.com/jdbc/jdbc-sample-code.htm

OpenAI. (2024). ChatGPT (versión GPT-5.1, modelo generativo de IA) [Modelo de lenguaje]. https://chat.openai.com/

Universidad Tecnológica Nacional, Cátedra Programación II. (2025). Módulos Audiovisuales: JDBC, Transacciones y Arquitectura por Capas. [Material de Clase]. Material no publicado, Tecnicatura Universitaria en Programación a Distancia.

Universidad Tecnológica Nacional, Cátedra Programación II. (2025). Material audiovisual sobre Arquitectura en Capas y Patrón DAO. [Archivo de Video de Clase]. Material no publicado, Tecnicatura Universitaria en Programación a Distancia.