UNIVERSIDAD TECNOLOGICA NACIONAL

Trabajo Final Integrador – Programación 2

Tecnicatura Universitaria en Programación

Universidad Tecnológica Nacional

Grupo: 73

Alumnos:

Matías Germán Corzo

Marcela Livio

Lucas Couchot

Matías Rodriguez

Contenido

Composición del equipo

Integrante	Rol
Matías Germán Corzo	Desarrollador
Marcela Livio	Analista Funcional
Lucas Couchot	Tester
Matías Rodriguez	Desarrollador

Dominio

Elegimos el modelo de relación unidireccional entre Empresa y DomicilioFiscal.

Este modelo nos pareció una buena forma de poner en práctica lo aprendido con respecto a relaciones, sobre todo porque creemos que es un problema muy común que puede llegar a presentarse en nuestra vida profesional.

Decisiones de diseño

Se optó por modelar dos tablas: empresa y domicilio_fiscal.

Los campos id los pensamos como auto incrementales para simplificar la asignación de los mismos.

Debido a que se optó por borrado lógico en lugar de físico se define el campo boolean eliminado en cada una de las tablas.

Finalmente, se optó por definir una foreign key (FK) en la tabla domicilio_fiscal referenciando al campo id de la tabla empresa para garantizar la integridad referencial y así no poder tener domicilios huérfanos en el sistema.

Este último punto podría ser discutible ya que al ser una relación uno a uno esta referencia podría modelarse en la tabla empresa directamente, sin embargo, nos pareció conveniente definirlo en la tabla domicilio_fiscal ya que esto permitiría a futuro tener múltiples domicilios vinculados a la misma empresa con cambios mínimos en caso de ser necesario.

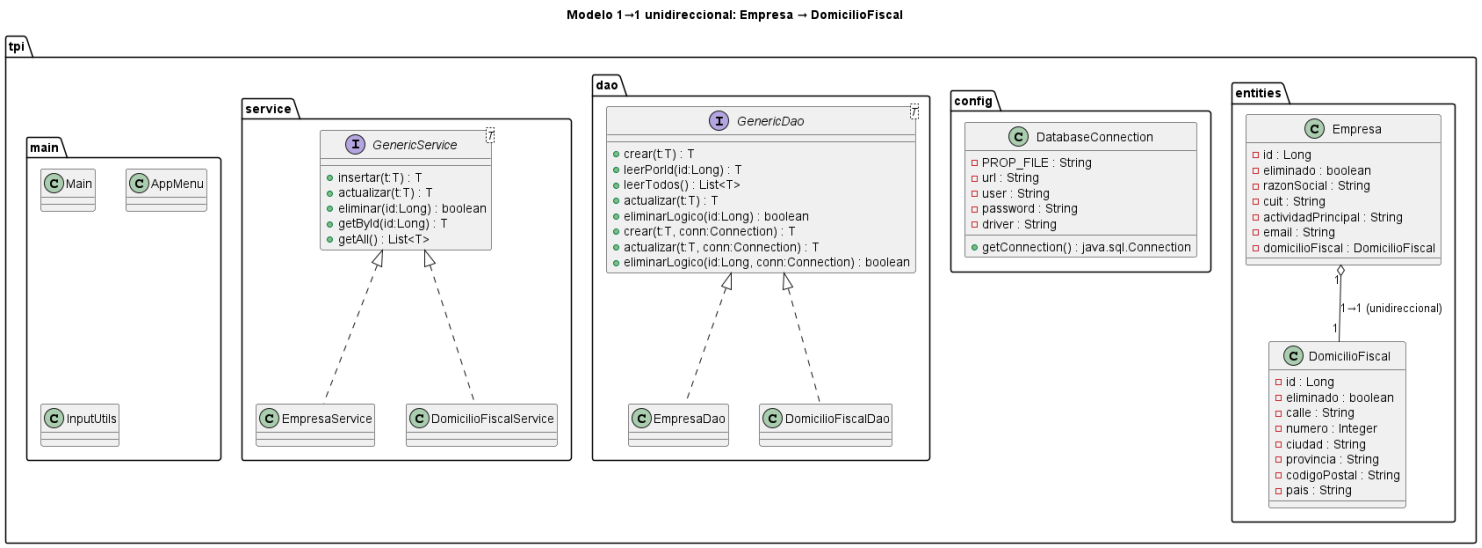
Las principales validaciones y reglas de negocio se verán reflejadas en la capa de servicios del proyecto.

Para resumir podemos listar las siguientes:

- Validaciones de nulidad sobre los campos obligatorios y que el número sea un número positivo válido al momento de crear o actualizar un Domicilio Fiscal.
- Validaciones de nulidad y longitud de CUIT al momento de crear o actualizar una Empresa.

Finalmente, presentamos el diagrama de modelado UML que planteamos con las diferentes clases.

En el mismo se puede observar en términos generales la completitud del sistema (entidades, DAOs, servicios, código de configuración y clases de menú e inicio).



Arquitectura

Se organizó el código en diferentes capas cada una con sus respectivas responsabilidades:

Capa	Responsabilidades
config	Clases de configuración
dao	Lógica de persistencia
entities	Entidades del modelo
main	Clase de inicio de ejecución y lógica de menú
service	Lógica de servicio

Esta arquitectura, si bien fue propuesta por la cátedra, nos pareció bastante clara y bien organizada como para diferenciar las diferentes responsabilidades de las diferentes partes del código.

Pruebas

A continuación listamos distintos escenarios de prueba y adjuntamos evidencia de los mismos:

Escenario 1

Al crear una nueva empresa de forma exitosa, debería reflejarse el cambio en el listado (Operación 2).

Listado previo a creación:

```
=== MENÚ  ===
1) Crear Empresa + Domicilio (transaccional)
2) Listar Empresas
3) Buscar Empresa por ID
4) Buscar Empresa por CUIT
5) Actualizar Empresa (+ Domicilio opcional)
6) Eliminar Empresa
7) Listar Domicilios Fiscales
8) Simular rollback (falla intencional)
0) Salir
Seleccione opción: 2

-- Listado de Empresas --
Empresa{id=1, eliminado=false, razonSocial='Acme SA', cuit='30-12345678-9', actividadPrincipal='Manufactura', email='contacto@acme.com', domici
Empresa{id=2, eliminado=false, razonSocial='Beta SRL', cuit='30-87654321-0', actividadPrincipal='Servicios', email='info@beta.com', domicilioFi
```

Creación:

```
Seleccione opción: 1

-- Alta Empresa + Domicilio --
Razón Social: Prueba SRL
CUIT: 20-12121212-5
Actividad principal (ENTER para dejar vacío): Importaciones
Email (ENTER para dejar vacío): pruebasrl@prueba.com
Calle: Simpre Viva
Número (ENTER para null): 123
Ciudad: CABA
Provincia: CABA
Código Postal (ENTER para dejar vacío): 1182
País: Argentina
Empresa creada con éxito. ID: 4
Empresa{id=4, eliminado=false, razonSocial='PRUEBA SRL', cuit='20-12121212-5',
```

Listado posterior:

```
Seleccione opción: 2

-- Listado de Empresas --
Empresa{id=1, eliminado=false, razonSocial='Acme SA', cuit='30-12345678-9', act
Empresa{id=2, eliminado=false, razonSocial='Beta SRL', cuit='30-87654321-0', a
Empresa{id=4, eliminado=false, razonSocial='PRUEBA SRL', cuit='20-12121212-5',
```

Escenario 2

Al buscar una empresa por su ID (Operación 3) se encuentra correctamente, caso contrario se retorna un error.

```
Seleccione opción: 3

-- Buscar por ID --
ID de Empresa (número): 4
Empresa{id=4, eliminado=false, razonSocial='PRUEBA SRL', cuit='20-12121212-5', act
```

```
Seleccione opción: 3

-- Buscar por ID --
ID de Empresa (número): 0
  * ID inválido. Intente de nuevo.
```

Escenario 3

Al buscar una empresa por CUIT (Operación 4) se retorna correctamente si existe, caso contrario se retorna un error.

```
Seleccione opción: 4

-- Buscar por CUIT --
CUIT: 30-12345678-9
Empresa{id=1, eliminado=false, razonSocial='Acme SA', cuit='30-12345678-9', act
```

```
Seleccione opción: 4

-- Buscar por CUIT --
CUIT: 12121212121
No existe empresa con CUIT 12121212121
```

Escenario 4

Al actualizar una empresa los cambios se actualizan correctamente, además al listar los datos de domicilio (Operación 7) se pueden verificar los cambios de forma exitosa:

```
-- Actualizar Empresa --
ID de Empresa a actualizar (número): 2
Nueva razón social (ENTER mantiene) (ENTER para dejar vacío): Nueva razón
Nuevo CUIT (ENTER mantiene) (ENTER para dejar vacío):
Nueva actividad principal (ENTER mantiene) (ENTER para dejar vacío): Nueva actividad
Nuevo email (ENTER mantiene) (ENTER para dejar vacío): nuevoemail@prueba.com
¿Actualizar domicilio fiscal también?
Confirmar [S/n]: S
Calle (ENTER mantiene) (ENTER para dejar vacío):
Número (ENTER mantiene) (ENTER para null): 1212
Ciudad (ENTER mantiene) (ENTER para dejar vacío): CABA
Provincia (ENTER mantiene) (ENTER para dejar vacío): CABA
Código Postal (ENTER mantiene) (ENTER para dejar vacío): 1212
País (ENTER mantiene) (ENTER para dejar vacío): Chile
Empresa actualizada con éxito.
Empresa{id=2, eliminado=false, razonSocial='NUEVA RAZÓN', cuit='30-87654321-0', actividadPrincipal='Nueva actividad', email='nuevoemail@prueba
```

```
Seleccione opción: 7

-- Listado de Domicilios Fiscales --
DomicilioFiscal{id=1, eliminado=false, calle='Av. Siempre Viva', numero=742, ciudad='Córdoba', provincia='Córdoba', codigoPostal='5000', pais='A
DomicilioFiscal{id=2, eliminado=false, calle='Calle Falsa', numero=1212, ciudad='CABA', provincia='CABA', codigoPostal='1212', pais='CHILE'}
DomicilioFiscal{id=3, eliminado=false, calle='Simpre Viva', numero=123, ciudad='CABA', provincia='CABA', codigoPostal='1182', pais='ARGENTINA'}
```

Escenario 5

Al ingresar la opción de salir (Operación 0), la aplicación finaliza correctamente:

```
=== MENÚ  ===
1) Crear Empresa + Domicilio (transaccional)
2) Listar Empresas
3) Buscar Empresa por ID
4) Buscar Empresa por CUIT
5) Actualizar Empresa (+ Domicilio opcional)
6) Eliminar Empresa
7) Listar Domicilios Fiscales
8) Simular rollback (falla intencional)
0) Salir
Seleccione opción: 0
Saliendo

Process finished with exit code 0
```

Conclusiones

Como conclusión principal no podemos olvidar destacar la importancia de realizar las tareas de diseño previo a ponerse a trabajar en el código.
Tanto el modelado UML como la planificación del modelo físico y arquitectura de la aplicación nos simplificaron muchísimo el desarrollo posterior, principalmente porque pudimos detectar a tiempo detalles que se nos podrían haber pasado y haber requerido mucho re-trabajo de no haber sido analizadas en su momento.

Otro aspecto a destacar es la importancia de realizar pruebas sobre el código luego de cada iteración o modificación en el mismo debido a que notamos la aparición de nuevos errores en partes donde el código se encontraba funcionando correctamente en ocasiones anteriores. Esto último podría mitigarse en gran medida con la implementación de código de pruebas automáticas como tests unitarios.

Uso de herramientas

- MySQL Workbench: Operaciones con la base de datos (creación, consultas, etc)
- Herramientas de IA (ChatGPT y Google Gemini): Intentamos no abusar de estas herramientas y las utilizamos principalmente para consultas generales sobre dudas específicas de Java y errores de bases de datos, sin alimentarla directamente con nuestro código.
- Netbeans, Intellij Idea y Visual Studio Code: Hicimos uso de distintos entornos de desarrollo tanto para implementar el proyecto como para ejecutar las pruebas.
- PlantUML: Se hizo uso de esta herramienta para generar el diagrama UML del proyecto.
- Github: Se hizo uso de Github para almacenar el repositorio del proyecto.