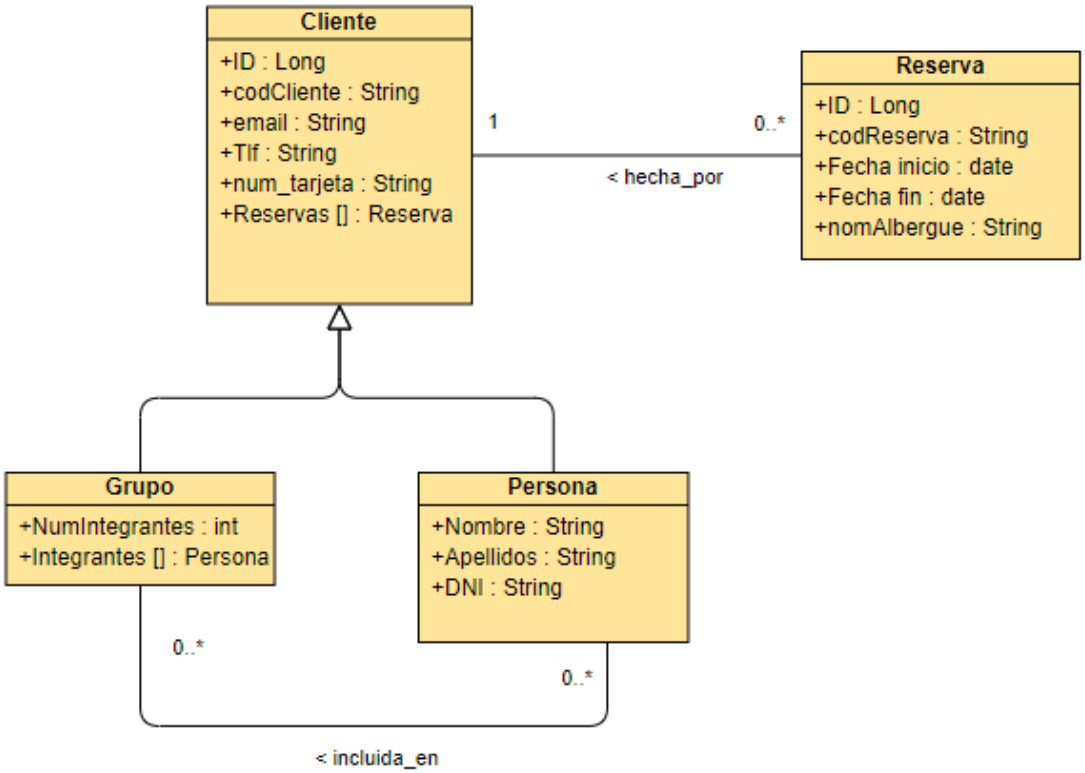
TT-ORM

Adrián Seco Fontecoba – a.seco@udc.es

Víctor Vázquez Mira – v.vazquez@udc.es

Diagrama de clases



Para nuestro modelo hemos creado dos asociaciones:

- Hecha_por: asociación bidireccional uno-muchos/muchos-uno
- Incluida_en: asociación unidireccional muchos a muchos

Casos de uso

1 Administrador: Alta nueva persona

Pantalla con formulario donde se introducen los datos de la nueva persona

- "Aceptar" y guardamos

1.1 Administrador: Añadir como integrante

- Pantalla con formulario donde se introducen los datos de la nueva persona

- Selección opción "añadir integrante"

- Pantalla con la lista de grupos creados donde se puede añadir

- Selección del grupo

- "Aceptar" y guardar

2 Administrador: Editar persona

- Pantalla con formulario para buscar la persona (nombre y apellidos, dni)

- Selección opción para modificar los datos de una persona de la lista

- Accedemos a un formulario de modificación de los datos (email, tlf, tarjeta, nombre, apellidos, dni)

- Modificamos y "Aceptar"

3 Administrador: Eliminar persona

- Pantalla con formulario para buscar la persona (nombre y apellidos, dni)

- Selección opción para eliminar persona

- Mensaje confirmación

- "Aceptar" y se elimina de la BD la persona, todas las reservas que tuviera asociadas y de todos los grupos en los que estuviera como integrante

4 Administrador: Listar reservas persona

- Pantalla con formulario para buscar la persona (nombre y apellidos, dni)

- Pantalla con los datos de esa persona en concreto y sus reservas ordenadas por Fecha inicio

4.1 Administrador: Ver reserva

- Pantalla con los datos de esa persona en concreto y sus reservas ordenadas por Fecha inicio

- Selección de una reserva

- Pantalla con los datos de la reserva

4.2 Administrador: Alta de una nueva Reserva

- Pantalla con los datos de esa persona en concreto y sus reservas ordenadas por Fecha inicio

- Selección de la opción "Nueva reserva"

- Pantalla con formulario para especificar los datos de una nueva reserva.

- "Aceptar" y guardamos

4.3 Administrador: Modificar reserva

- Pantalla con los datos de esa persona en concreto y sus reservas ordenadas por Fecha inicio

- Selección opción para editar la reserva de la lista

- Pantalla con un formulario para editar los datos de la reserva

- Modificamos y guardamos

4.4 Administrador: Eliminar reserva

- Pantalla con los datos de esa persona en concreto y sus reservas ordenadas por Fecha inicio

- Selección opción para eliminar la reserva de la lista

- Mensaje confirmación

- Eliminación de la reserva de la BD

5. Administrador: Alta nuevo grupo

- Pantalla con formulario donde se introducen los datos del nuevo grupo

- "Aceptar" y guardamos

5.1 Administrador: Adición de los integrantes

- Pantalla con formulario donde se introducen los datos del nuevo grupo

- Buscador por dni de las personas creadas

- Selección del resultado de la búsqueda para añadirlo a los integrantes del grupo

- En caso de no haber resultados, selección opción para añadir una nueva persona (Caso de uso 1)

- "Aceptar" y guardar

6. Administrador: Listar grupos

- Pantalla con la lista de todos los grupos creados

6.1 Administrador: Ver grupo

- Pantalla con la lista de todos los grupos creados

- Selección del grupo

- Pantalla con los datos del grupo seleccionado

6.1.2 Administrador: Listar integrantes

- Pantalla con los datos del grupo seleccionado

- Selección de la pestaña integrantes

- Vista de la lista de los integrantes del grupo

6.1.2.1 Administrador: Añadir integrante

- Vista de la lista de los integrantes del grupo

- Selección opción "Añadir"

- Dispara caso de uso 5.1

6.1.2.2 Administrador: Eliminar integrante

- Vista de la lista de los integrantes del grupo

- Selección opción "eliminar"

- Borra al integrante de la colección

6.2 Administrador: Editar grupo

- Pantalla con la lista de todos los grupos creados

- Selección de la opción editar

- Pantalla con el formulario para editar los datos (email, tlf, tarjeta, intergantes)

6.3 Administrador: Eliminar grupo

- Pantalla con la lista de todos los grupos creados

- Selección opción eliminar

- Se elimina el grupo

6.4 Administrador: Listar reservas del grupo

- Dispara el caso de uso 4

7. Administrador: Listar reservas

- Pantalla con una lista con todas las reservas ordenadas por fecha de inicio

7.1 Administrador: Filtro reservas por fecha concreta

- Pantalla con buscador donde se indicará la fecha de inicio

- Lista con las reservas que tienen como fecha de inicio la indicada

7.2 Administrador: Filtro reservas por "nombre albergue"

- Pantalla con buscador donde se indicará el nombre del albergue
- Lista con todas las reservas asociadas a ese albergue

Propagación de las operaciones

REMOVE	Cliente		Reserva
	Integrantes [] P (mappedBy)	**Reservas [] I (mappedBy)**	**Cliente P**
3 (Eliminar persona)	No (1)	Si (2)	
4.4 (Eliminar reserva)			No (3)
6.1.2.2 (Eliminar integrante)		No (4)	
6.3 (Eliminar grupo)	No (5)	Si (6)	
ELECCIÓN FINAL	**No**	**Si**	**No**

Justificación de las decisiones adoptadas para la operación **REMOVE**

(1) : Cuando eliminamos una persona no tiene sentido que desaparezca de los integrantes de un grupo, pues el grupo seguirá existiendo. Como estamos del lado inverso, eliminar una persona no afectará a la tabla de ***grupo***, que seguirá manteniendo sus integrantes. Por el contrario, si se eliminará su entrada de la tabla intermedia ***composición_grupos***. Por ello, **NO** nos interesa propagar la operación *REMOVE.*

(2) : Cuando borramos una persona, no tiene sentido que las reservas asociadas a esa persona permanezcan en la base de datos. De esta forma, **SI** nos interesa propagar la operación *REMOVE*. Si borramos la persona C1, su fila de la tabla ***persona*** desaparece. Al eliminar las reservas R1 y R2, sus filas desaparecen de la tabla ***reserva*** y con ellas las relaciones existentes con la persona C1 (No hay violación de integridad referencial).

(3) : Eliminar una reserva no implica que se elimine el cliente (persona o grupo) que la haya realizado, pues podría querer hacer más reservas en otro momento. Teniendo esto en cuenta, **NO** nos interesa propagar la operación *REMOVE*.

(4) : Al eliminar un integrante no tiene sentido que se elimine las reservas de la persona a la que hacer referencia ya que estamos editando un atributo colección de la tabla ***grupo***. Por ello, **NO** nos interesa propagar la operación *REMOVE*. La persona a la que hace referencia el atributo integrante puede querer realizar sus reservas individuales.

(5) : La eliminación de un grupo no debe implicar el borrado de las personas que forman los integrantes, pues estas podrían querer hacer reservas de forma individual. Dado que nos encontramos del lado ***propietario*** de la relación (relación muchos a muchos unidireccional persona -> grupo), **NO** queremos que se propague.

(6) : En este caso pasa lo mismo que en el apartado (2). Un grupo tendrá asociadas sus reservas. Si eliminamos un grupo, sus reservas quedarán desreferenciadas violando la restricción de integridad referencial. Por ello, **SI** nos interesa propagar la operación *REMOVE*.

PERSIST	Cliente		Reserva
	Integrantes [] P (mappedBy)	**Reservas [] I (mappedBy)**	**Cliente P**
1 (Alta persona) **5.1 (Adición integrantes)** **6.1.2.1 (Añadir integrante)**		No (1)	
4.2 (Alta reserva)			No (2)
5 (Alta grupo)	No (3)		
ELECCIÓN FINAL	**No**	**No**	**No**

Justificación de las decisiones adoptadas para la operación **PERSIST**

(1) : Cuando damos de alta una persona, no implica dar de alta sus reservas. Primero debe estar registrada la persona para poder realizar las reservas. Por ello, la propagación **NO** tiene sentido.

(2) : Al dar de alta una nueva reserva, accedemos previamente al cliente que la va a realizar. Por ello **NO** es necesario propagar la operación *PERSIST*. El cliente debe estar registrado previamente para poder realizar la reserva.

(3) : Para dar de alta un grupo, previamente han tenido que darse de alta los integrantes que lo compondrán. En este caso, **NO** se propagará la operación.

MERGE	Cliente		Reserva
	Integrantes [] P (mappedBy)	**Reservas [] I (mappedBy)**	**Cliente P**
2 (Editar persona)	No (1)	No (2)	
4.3 (Modificar reserva)			No (3)
6.2 (Editar grupo)		No (4)	
ELECCIÓN FINAL	**No**	**No**	**No**

Justificación de las decisiones adoptadas para la operación **PERSIST**

(1) : Al modificar una persona podría pensarse en la necesidad de propagar la operación a la colección de ***Integrantes*** de la tabla ***grupo***. Sin embargo, esto no es así puesto que nos encontramos en el lado propietario de la asociación unidireccional entre ***grupo -> persona***, por ello el controlador de persistencia ya detectará los cambios actuando en consecuencia. Es por ello que **NO** necesitamos propagar la operación *MERGE*.

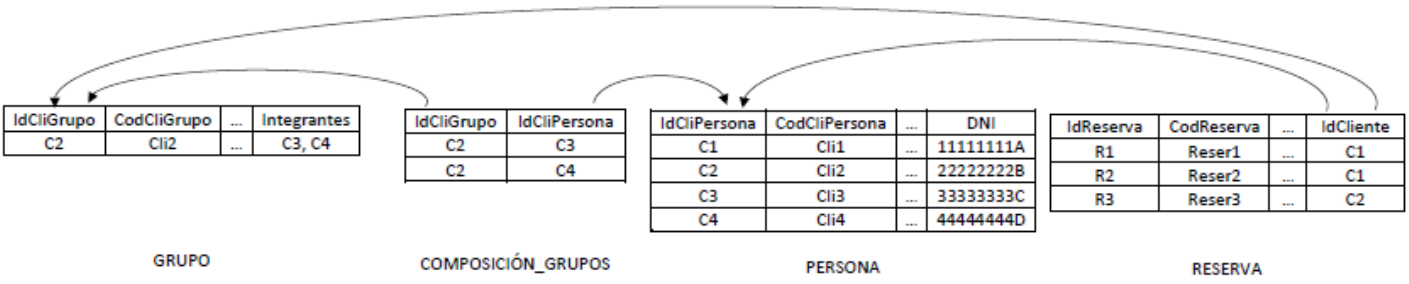
(2) : Para modificar una persona, recogemos el objeto desligado C con su nuevo estado y aplicamos *merge* sobre el para crear en el Control de Persistencia una copia C' de C que será sincronizada con la base de datos. En ningún momento se permite la modificación de la lista de reservas ni de su información relativa. Por ello, la propagación de la operación *MERGE* **NO** tiene sentido en este caso.

(3) : Para modificar una reserva, recogemos el objeto desligado R con su nuevo estado y aplicamos *merge* sobre el para crear en el Control de Persistencia una copia R' de R que será sincronizada con la base de datos. En ningún momento se permite la modificación del cliente vinculado a esa reserva. Por ello, la propagación de la operación *MERGE* **NO** tiene sentido en este caso.

(4) : Para modificar un grupo, recogemos el objeto desligado C con su nuevo estado y aplicamos *merge* sobre el para crear en el Control de Persistencia una copia C' de C que será sincronizada con la base de datos. En ningún momento se permite la modificación de la lista de reservas ni de su información relativa. Por ello, la propagación de la operación *MERGE* **NO** tiene sentido en este caso.

Análisis aislado de las pantallas

Caso de uso	Objetos necesarios en memoria para cubrir los datos de la pantalla	Cliente		Reserva
		Integrantes [] P (mappedBy)	Reservas [] I (mappedBy)	Cliente P
1 (Alta persona)	-			
1.1 (Añadir como integrante)	Persona única Todos los grupos	LAZY (1)	LAZY (2)	
2 (Editar persona)	Persona única	LAZY (3)	LAZY (4)	
3 (Eliminar persona)	Persona única Colección de sus reservas	LAZY (5)	LAZY (6)	LAZY (7)
4 (Listar reservas persona)	Persona única: C Colección de sus reservas	LAZY (8)	EAGER (9)	LAZY (10)
4.1 (Ver reserva)	Persona o grupo únicos Reserva única	LAZY (11)	LAZY (12)	EAGER (13)
4.2 (Alta reserva)	Grupo o persona únicas	LAZY (14)	LAZY (15)	
4.3 (Modificar reserva)	Persona o grupo únicos Reserva asociada	LAZY (16)	LAZY (17)	EAGER (18)
4.4 (Eliminar reserva)	Persona o grupo únicos Reserva asociada	LAZY (19)	EAGER (20)	LAZY (21)
5 (Alta grupo)	-			
5.1 (Añadir integrantes) **6.1.2.1 (Añadir integrante)**	Grupo único Colección de integrantes	LAZY (22)	LAZY (23)	
6 (Listar grupos)	Todos los grupos: C1, C2, …	LAZY (24)	LAZY (25)	
6.1 (Ver grupo)	Grupo único	LAZY (26)	LAZY (27)	
6.1.2 (Listar integrantes)	Grupo único Colección de todos sus integrantes	EAGER (28)	LAZY (29)	
6.1.2.2 (Eliminar integrante)	Grupo único Colección de sus integrantes	LAZY (30)	LAZY (31)	
6.2 (Editar grupo)	Grupo único Colección de sus integrantes	EAGER (32)	LAZY (33)	
6.3 (Eliminar grupo)	Todos los grupos	LAZY (34)	LAZY (35)	LAZY (36)
6.4 (Listar reservas grupo)	Grupo único Colección de sus reservas	LAZY (37)	EAGER (38)	LAZY (39)
7 (Listar reservas)	Todas las reservas			EAGER (40)
7.1 (Filtro reservas por fecha)	Todas las reservas que cumplan la condición			LAZY (41)
7.2 (Filtro reservas por nombre albergue)	Todas las reservas que cumplan la condición			LAZY (42)
ELECCIÓN FINAL		**LAZY**	**LAZY**	**EAGER**



1 Alta persona

La pantalla se corresponde con un formulario vacío. Por ello, NO es necesario cargar ningún tipo de dato. (Carga ***NO NECESARIA***)

1.1 Añadir como integrante

(1) *(Integrantes [] -> lazy):* En esta pantalla cargamos una única persona y todos los grupos existentes en la base de datos. En la pantalla solo mostramos los atributos básicos del estado de persona. También debemos cargar **todos** los grupos existentes en la base de datos para poder seleccionar en cual se insertará como nuevo integrante. Dado que vamos a hacer una modificación en la colección de integrantes esta colección debería estar en memoria. Pero habría que cargar la colección de todos los grupos existentes ya que nos sabemos en cual se va a insertar. Sin embargo, se encuentra del lado propietario por lo que se realiza una carga EAGER por defecto. (Carga ***LAZY***)

(2) *(reservas [] -> lazy):* En esta pantalla **mostramos** únicamente atributos básicos del estado de la persona seleccionada y de los grupos existentes. No es necesario mostrar ni modificar NINGUNA información relativa a las reservas de la persona ni de los grupos. Tampoco vamos a hacer cambios ni modificar información básica de las reservas. Por tanto, no es preciso acceder a esta colección en ningún caso. (Carga ***LAZY***)

2 Editar persona

(3) *(Integrantes [] -> lazy):* En esta pantalla mostramos únicamente **nombre, apellidos, dni, email, teléfono** y **tarjeta de crédito** de una persona concreta, que son atributos básicos de su estado y los únicos que se pueden modificar. Modificamos el estado de la persona e invocamos la operación *merge*. No es necesario mostrar la colección de integrantes en la pantalla. Tampoco se realizarán cambios ni modificaciones en ella, por lo que no es preciso acceder a esta colección. (Carga ***LAZY***)

(4) *(reservas [] -> lazy):* En esta pantalla mostramos únicamente **nombre, apellidos, dni, email, teléfono** y **tarjeta de crédito** de una persona concreta, que son atributos básicos de su estado y los únicos que se pueden modificar. No es necesario mostrar la colección de reservas en la pantalla. Tampoco se realizarán cambios ni modificaciones en ella, por lo que no es preciso acceder a esta colección. (Carga ***LAZY***)

3 Eliminar persona

(5) *(integrantes [] -> lazy):* En esta pantalla solo mostramos atributos básicos del estado de una persona. No es necesario mostrar la colección de integrantes en la pantalla. Tampoco se

realizarán cambios ni modificaciones en ella, por lo que no es necesario acceder a esta colección. (Carga ***LAZY***)

(6) *(reservas [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos del estado de una persona. Aplicaremos la operación *remove* que se propagará a todas sus reservas.

Consideraciones:

- Propagación: Existe. Cunado eliminamos una persona, TODAS sus reservas deben ser eliminadas, propagando la operación *remove*. Para poder hacerlo la colección debe estar cargada en memoria. Cargarla de forma anticipada cuando carguemos la persona **sería beneficioso**.
- Eliminación de las asociaciones: Propagaremos el borrado desde la persona a todas sus reservas. Al eliminar las reservas de la base de datos desaparecerán las claves foráneas a dicha persona.

(7) *(cliente - > lazy):* Cuando apliquemos *remove* sobre la persona, se cargarán en memoria su colección de revistas para propagar sobre ellas la operación. No necesitamos que se carguen anticipadamente las referencias a persona desde sus reservas porque la persona **ya se encuentra cargada en memoria** y además vamos a eliminar esas reservas. (Carga ***LAZY***)

4 Listar reservas persona

(8) *(Integrantes [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos del estado de la persona. No es necesario mostrar NINGUNA información básica de los integrantes, por lo que no es necesario acceder a esa colección. (Carga ***LAZY***)

(9) *(Reservas [] -> eager):* En esta pantalla solo **mostramos** información. Aparece el **nombre completo, dni, email** y **teléfono** de la persona seleccionada (C), que son atributos básicos de su estado, pero TAMBIÉN es necesario mostrar el código y la fecha de inicio de CADA reserva R1, R2… que la persona ha realizado y que son atributos básicos del estado de R1, R2… Cargar la colección de las reservas al cargar la persona **puede suponer una ventaja** en este caso. (Carga ***EAGER***).

(10) *(Cliente -> lazy):* Debemos analizar este caso porque estamos suponiendo una carga EAGER de las reservas R1, R2… de la persona C. Teniendo esto en cuenta, ya estaríamos cargando C en memoria por lo que **todas** las reservas serán de la persona C. Por tanto, es **innecesario** inicializar las referencias desde R1, R2… hacia C. (Carga ***LAZY***)

4.1 Ver reserva

(11) *(Integrantes [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos del estado de la persona o grupo seleccionado (C). No es necesario mostrar NINGUNA información básica de los integrantes, por lo que no es necesario acceder a esa colección. (Carga ***LAZY***)

(12) *(Reservas [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos de la persona o grupo seleccionado (C), y datos básicos de UNA de sus reservas (R). Si establecemos una carga EAGER estaríamos cargando TODAS las reservas de esa persona/grupo. Como solo queremos una única reserva, no nos interesa. (Carga ***LAZY***)

(13) *(Cliente -> eager):* Al descartar la carga EAGER para Cliente.reserva, y dado que analizamos las pantallas de forma aislada de las demás, consideramos la opción alternativa cargando la reserva R en memoria antes que la persona/grupo. De esta forma activamos una carga EAGER

de Reserva.cliente consiguiendo cargar simultáneamente tanto la reserva R como su único cliente asociado (persona/grupo) C. (Carga ***EAGER***)

4.2 Alta reserva

(14) *(integrantes [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos de la persona o grupo seleccionado (C). No es necesario mostrar ni modificar NINGUNA información relativa a los integrantes del grupo. Tampoco vamos a realizar ningún cambio en esta colección ni modificar información básica en ellos. Por lo tanto, no es preciso acceder a esta colección en ningún caso. (Carga ***LAZY***)

(15) *(reservas [] -> lazy):* En esta pantalla cargamos una única persona/grupo. En la pantalla se mostrarán únicamente atributos básicos de su estado. Además, creamos un objeto que representará la nueva reserva R, que asociaremos con la persona/grupo, e invocaremos la operación *persist*.

Consideraciones:

- No es necesario mostrar en la pantalla NINGUNA información relativa a la colección de reservas de la persona/grupo seleccionada. Por ello, una carga EAGER no es necesaria.
- Como el nuevo objeto reserva está en el lado propietario, incluirá una referencia a la persona/grupo (clave foránea). (Carga ***LAZY***)

4.3 Modificar reserva

(16) *(Integrantes [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos de la persona o grupo seleccionado y datos básicos de UNA de sus reservas (R). No es necesario mostrar ni modificar NINGUNA información relativa a los integrantes del grupo. Tampoco vamos a hacer cambios ni modificaciones en esta colección. Por tanto, no es preciso acceder a esta colección en ningún caso. (Carga ***LAZY***)

(17) *(reservas [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos de la persona o grupo seleccionado (C), y datos básicos de UNA de sus reservas (R). Si establecemos una carga EAGER estaríamos cargando TODAS las reservas de esa persona/grupo. Como solo queremos una única reserva, no nos interesa. (Carga ***LAZY***)

(18) *(cliente - > eager):* Descartada la carga EAGER para Cliente.reserva, y dado que analizamos esta pantalla de forma aislada a las demás, podemos considerar la opción alternativa de cargar la reserva R en memoria antes que la persona/grupo. Activando la carga EAGER de Reserva.cliente conseguimos cargar simultáneamente tanto la reserva como la persona/grupo asociado.

Como vamos a aplicar una operación *merge* sobre reserva debemos considerar:

- Propagación: No hay propagación del *merge* desde reserva a cliente (persona/grupo). Por lo tanto, no es necesario tener inicializada la referencia desde el cliente a la reserva para poder accede a dicha reserva y propagar la operación. Por ello, esto no sirve como argumento para activar la carga EAGER.

- Cambios en la asociación: La reserva no va a cambiar de cliente (persona/grupo). Por lo tanto, no es necesario tener inicializada la referencia desde reserva a cliente para poder hacer que apunte a otra persona/grupo.

Las consideraciones no nos proporcionan argumentos favorables para la elección de la carga EAGER. Pero esta carga tampoco sería perjudicial en ambos casos, así que la elegimos porque nos interesa cargar simultáneamente reserva y persona/grupo. (Carga ***EAGER***)

4.4 Eliminar reserva

(19) *(Integrantes [] -> lazy):* En esta pantalla mostramos únicamente atributos básicos de la persona o grupo seleccionado y datos básicos de UNA de sus reservas (R). No es necesario mostrar ni modificar NINGUNA información relativa a los integrantes del grupo. Tampoco vamos a hacer cambios ni modificaciones en esta colección. Por tanto, no es preciso acceder a esta colección en ningún caso. (Carga ***LAZY***)

(20) *(reservas [] -> eager):* En esta pantalla mostramos únicamente atributos básicos de la persona o grupo seleccionado y datos básicos de UNA de sus reservas (R). Cargar la colección de reservas **podría suponer una ventaja** en este caso.

También debemos tener en cuenta que vamos a eliminar una reserva (R) de una persona/grupo. Esto significa que eliminar la reserva de la colección de reservas de ese cliente por lo que tener inicializada la colección supondría una ventaja. Sin embargo, la colección de reservas se encuentra en el **lado inverso** de la asociación por lo que el cambio no tendría efecto sobre la base de datos. (Carga ***EAGER***)

(21) *(Cliente -> lazy):* En esta pantalla únicamente mostramos datos básicos del estado de una reserva (R) y aplicaremos la operación *remove* sobre él.

Consideraciones:

- Propagación: No necesitamos propagación. Cuando eliminamos una reserva no debe eliminarse la información de la persona/grupo asociado. Estos clientes pueden querer realizar más reservas en un futuro. Por tanto, **no es necesario** acceder a clientes con esa finalidad. No se justifica la carga EAGER.
- Asociaciones: Si debe ser eliminada la asociación de la reserva con ese cliente (persona/grupo). Como reserva es el lado propietario de la asociación, el proveedor de persistencia eliminará automáticamente las claves foráneas entre reservas y cliente. No necesitamos acceder al otro extremo para eliminar la reserva de la colección de reservas del cliente (persona/grupo). No se justifica la carga EAGER.

(Carga ***LAZY***)

5 Alta grupo

La pantalla se corresponde con un formulario vacío. Por ello, NO es necesario cargar ningún tipo de dato. (Carga ***NO NECESARIA***)

5.1 Añadir integrantes / 6.1.2.1 Añadir integrante

(22) *(integrantes [] -> lazy):* En esta pantalla cargamos un único grupo. En la pantalla solo mostramos los atributos básicos de su estado. También debemos cargar **todas** las personas existentes en la base de datos para poder seleccionar las que formarán parte de los integrantes del grupo. Incluiremos en su colección de integrantes referencias a las personas que conforman el grupo. Como se encuentra del lado propietario (asociación unidireccional muchos a muchos persona -> grupo), el proveedor de persistencia creará las referencias necesarias. (Carga ***LAZY***)

(23) *(reservas [] -> lazy):* En esta pantalla **mostramos** únicamente atributos básicos del estado del grupo seleccionado. No es necesario mostrar ni modificar NINGUNA información relativa a las reservas del grupo. Tampoco vamos a hacer cambios ni modificar información básica de las reservas. Por tanto, no es preciso acceder a esta colección en ningún caso. (Carga ***LAZY***)

6 Listar grupos

(24) *(integrantes [] -> lazy):* En esta pantalla solo se **muestra** información. Aparecen los atributos básicos de su estado de todos los grupos (C1, C2, …). No es necesario mostrar NINGUNA información relativa a los integrantes de dichos grupos, por ello, NO es necesario acceder a esta colección en ningún caso. (Carga ***LAZY***)

(25) *(reservas []-> lazy):* En esta pantalla solo se **muestra** información. Aparecen los atributos básicos de su estado de todos los grupos (C1, C2, …). No es necesario mostrar NINGUNA información relativa a los integrantes de dichos grupos, por ello, NO es necesario acceder a esta colección en ningún caso. (Carga ***LAZY***)

6.1 Ver grupo

(26) *(integrantes [] -> lazy):* En esta pantalla solo se **muestra** información. Aparecen los atributos básicos de su estado del grupo C seleccionado. No es necesario mostrar NINGUNA información relativa a los integrantes de dichos grupos, por ello, NO es necesario acceder a esta colección. (Carga ***LAZY***)

(27) *(reservas [] -> lazy):* En esta pantalla solo se **muestra** información. Aparecen los atributos básicos de su estado del grupo C seleccionado. No es necesario mostrar NINGUNA información relativa a las reservas de dichos grupos, por ello, NO es necesario acceder a esta colección. (Carga ***LAZY***)

6.1.2 Listar integrantes

(28) *(integrantes [] -> eager):* En esta pantalla solo se **muestra** información. Aparecen los atributos básicos de su estado del grupo C seleccionado, pero TAMBIÉN es necesario mostrar el **nombre completo** de CADA integrante perteneciente al grupo y que son atributos básicos del estado de dichos integrantes. Cargar la colección de los integrantes **puede suponer una ventaja** en este caso (e.g añadir un nuevo integrante). (Carga ***EAGER***)

(29) *(reservas [] -> lazy):* En esta pantalla solo se **muestra** información. Aparecen los atributos básicos de su estado del grupo C seleccionado y de su colección de integrantes. No es necesario mostrar NINGUNA información relativa a las reservas de dichos grupos y/o integrantes, por ello, NO es necesario acceder a esta colección. (Carga ***LAZY***)

6.1.2.2 Eliminar integrante

(30) *(Integrantes [] -> lazy):* Debemos analizar este caso porque suponemos carga EAGER de los integrantes de un grupo. Mas tarde elegiremos uno y aplicamos la operación *remove*.

Consideraciones:

- Propagación: No hay propagación del *remove* desde el grupo a la persona, así que no necesitamos tener una referencia para implementar la propagación. No se justifica EAGER

- Asociaciones: Estamos en el lado propietario. Cuando eliminamos un integrante de la colección desaparecerá con él la referencia a la tabla intermedia entre grupo-persona. No se justifica la carga EAGER

(31) *(reservas [] -> lazy):* En esta pantalla mostramos únicamente información básica del estado de un grupo concreto. No es necesario mostrar ni modificar NINGUNA información relativa a las reservas. Tampoco vamos a hacer cambios ni modificaciones en esta colección. Por lo tanto, no es necesario acceder a esta colección en ningún caso.

6.2 Editar grupo

(32) *(integrantes [] -> eager):* En esta pantalla mostramos los datos básicos de un grupo, pero también necesitamos saber que integrantes lo componen. En este caso podemos modificar información básica del grupo, pero también su colección de integrantes (añadiendo o eliminando integrantes). Por ello se aplicará una operación *merge* sobre el grupo.

Consideraciones:

- La carga EAGER es beneficiosa porque el grupo ya viene con su colección de integrantes inicializada.
- Hay que tener en cuenta que necesitaremos tener todas las personas cargadas en caso de querer añadir más integrantes. Amortizamos el uso del EAGER
- Asociaciones: Cualquier cambio en las personas asociadas como integrantes deberían reflejarse en esta colección. En este caso estamos en el lado propietario por lo que los cambios llegarán a la base de datos. Tener precargados los datos de persona nos será útil en caso de añadir integrantes. Justifica el uso del EAGER
- Propagación: No hay propagación desde grupo a persona, por lo que no vamos a modificar información propia de persona. En este caso no se justifica la carga EAGER. Sin embargo, hacer la carga anticipada no representa un problema y nos brinda más beneficios que inconvenientes, por lo que elegimos carga ***EAGER***.

(33) *(reservas [] -> lazy):* En esta pantalla mostramos los datos básicos de un grupo, pero también necesitamos saber que integrantes lo componen. No es necesario mostrar ni modificar NINGUNA información relativa a las reservas del grupo seleccionado. Tampoco se realizarán cambios ni modificaciones de información básica de las mismas. Por tanto, no es necesario acceder a esta colección en ningún caso. (Carga ***LAZY***)

6.3 Eliminar grupo

(34) *(integrantes -> lazy):* En esta pantalla mostramos únicamente atributos básicos del estado de un grupo. Aplicaremos la operación *remove* para eliminar el grupo. Al eliminar un grupo se debe eliminar su colección de integrantes y sus referencias en la tabla intermedia entre grupo-persona. Como nos encontramos del lado propietario estas referencias las manejará el controlador de persistencia eliminándolas de la tabla intermedia. Pensar en una carga EAGER, en este caso supondría cargar todos los integrantes de todos los grupos. **No compensa**. Cargaremos todos los grupos cuando se presenten en la pantalla y cunado se elija un grupo para eliminar, el proveedor de persistencia **recuperará únicamente** (y en diferido) la colección de integrantes de ese grupo. (Carga ***LAZY***)

(35) *(reservas > lazy):* En esta pantalla mostramos únicamente atributos básicos del estado de un grupo. Aplicaremos la operación *remove* que se propagará a todas sus reservas.

Consideraciones:

- Propagación: Existe. Cunado eliminamos un grupo, TODAS sus reservas deben ser eliminadas, propagando la operación *remove*. Para poder hacerlo la colección debe estar cargada en memoria. Cargarla de forma anticipada cuando carguemos la persona **sería beneficioso**.
  El problema en ese caso está en que si decidimos una carga EAGER estaríamos cargando todas las reservas de todos los grupos. **No compensa.** Cargaremos todos los grupos sin la colección reservas, y cuando se elija un grupo para eliminar, el proveedor de persistencia **recuperará únicamente** (y en diferido) la colección de reservas haciendo un nuevo acceso a la base de datos, accederá a ellos para eliminarlos propagando la operación.
- Eliminación de las asociaciones: Propagaremos el borrado desde el grupo a todas sus reservas. Al eliminar las reservas de la base de datos desaparecerán las claves foráneas a dicho grupo. La propagación es la solución, por lo que aplicamos el mismo razonamiento que en el apartado anterior.

(Carga ***LAZY***)

(36) *(cliente -> lazy):* Cuando apliquemos *remove* sobre el grupo, se cargarán en memoria su colección de revistas para propagar sobre ellas la operación. No necesitamos que se carguen anticipadamente las referencias a grupo desde sus reservas porque el grupo **ya se encuentra cargado en memoria** y además vamos a eliminar esas reservas. (Carga ***LAZY***)

6.4 Listar reservas grupo

(37) *(Integrantes [] -> lazy):* En esta pantalla solo **mostramos** información. Aparece únicamente el **código, email** y **teléfono** asociados al grupo seleccionado (C), que son atributos básicos de su estado. No es necesario mostrar NINGUNA información relativa a los integrantes del grupo, por lo que no es necesario acceder a esa colección en ningún caso. (Carga ***LAZY***)

(38) *(Reservas [] -> eager):* En esta pantalla solo **mostramos** información. Aparece únicamente el **código, email** y **teléfono** asociados al grupo seleccionado (C), que son atributos básicos de su estado, pero TAMBIÉN es necesario mostrar el **código** y la **fecha de inicio** de CADA reserva R1, R2… que la persona ha realizado y que son atributos básicos del estado de R1, R2… Cargar la colección de las reservas al cargar la persona **puede suponer una ventaja** en este caso. (Carga ***EAGER***).

(39) *(Cliente-> lazy):* Debemos analizar este caso porque estamos suponiendo una carga EAGER de las reservas R1, R2… del grupo C. Teniendo esto en cuenta, ya estaríamos cargando C en memoria por lo que **todas** las reservas serán de ese grupo C. Por tanto, es **innecesario** inicializar las referencias desde R1, R2… hacia C. (Carga ***LAZY***)

7 Listar reservas

(40) *(cliente -> eager):* En esta pantalla solo **mostramos** información. Aparecen únicamente atributos básicos del estado de reserva. Sin embargo, en este caso nos interesa traerlo todo pues estamos listando TODAS las reservas de TODOS los clientes. (Carga ***EAGER***)

7.1 Filtro reservas por fecha

(41) *(cliente -> lazy):* En esta pantalla solo **mostramos** información. Aparecen únicamente atributos básicos del estado de reserva. Sin embargo, a diferencia del caso *7 Listar reservas*, no

nos interesa cargar todas las reservas, sino solo las que cumplan la condición del filtro. (Carga ***LAZY***)

7.2 Filtro reservas por nombre albergue

(42) *(cliente -> lazy):* En esta pantalla solo **mostramos** información. Aparecen únicamente atributos básicos del estado de reserva. Sin embargo, a diferencia del caso *7 Listar reservas*, no nos interesa cargar todas las reservas, sino solo las que cumplan la condición del filtro. (Carga ***LAZY***)

Análisis de conflictos

1 Cliente.integrantes

En los casos en los que una carga EAGER es favorable son: (27) y (31). Analizaremos si alguno de los razonamientos correspondientes a estos casos queda anulados ya sea por el flujo de pantallas o por los razonamientos que aconsejan LAZY.

(27) vs (23): Para llegar a la pantalla del caso de uso de (27), *Listar integrantes*, tenemos que haber pasado previamente por la pantalla del caso de uso de (23), *Listar grupos*. Para cargar anticipadamente los integrantes de un **único** grupo, cargaremos los de **todos** ellos. Consideramos que el beneficio indicado en (27) no compensa el inconveniente de cargar dos veces los datos en memoria.

(31) vs (23): La pantalla del caso de uso de (31), *Editar grupo*, es la misma que la del caso de (23), *Listar grupos*. Por ello aplicamos el mismo razonamiento obteniendo el mismo resultado: renunciar a la carga EAGER.

Finalmente, decidimos aplicar una carga **LAZY**

2 Cliente.reservas

En los casos de uso en los que una carga EAGER es favorable son: (8), (19) y (37). Analizaremos si alguno de los razonamientos correspondientes a estos casos queda anulados ya sea por el flujo de pantallas o por los razonamientos que aconsejan LAZY.

(8) vs (4): Para llegar a la pantalla del caso de uso de (8), *Listar reservas persona*, tenemos que haber pasado previamente por la pantalla del caso de uso de (4), *Editar persona*. Al lanzar el caso de uso de *Editar persona*, no necesitamos cargar la colección de reservas, ya que ni se visualizan, ni se modifican. Sin embargo, si las necesitamos recuperar para poder listar las reservas individuales. El problema está en que con una carga EAGER nos traeríamos todas las reservas de todas las personas. Renunciamos a la carga EAGER.

(19) vs (8): Para llegar a la pantalla del caso de uso de (19), *Eliminar reserva*, es necesario, previamente, haber pasado por la pantalla del caso de uso de (8). Como hemos comentado en el apartado anterior, hacer un EAGER para recuperar la colección de revistas de una persona implica que se traerán todas las reservas de todas las personas por lo que renunciamos a la carga EAGER.

(37) vs (24): Para llegar a la pantalla del caso de uso de (37), *Listar reservas grupo*, primero necesitamos pasar por la pantalla del caso de uso de (26), *Ver grupo*. Al visualizar los grupos no necesitamos cargar la colección de las reservas pues no se hace nada con esos datos en esta pantalla. Sin embargo, como previamente se han listado los grupos, ya disponemos de los datos en memoria. Renunciamos a la carga EAGER.

Finalmente elegimos aplicar una carga **LAZY**.

3 Reserva.cliente

En los casos de uso en los que una carga EAGER es favorable son: (12), (17) y (39). Analizaremos si alguno de los razonamientos correspondientes a estos casos queda anulados ya sea por el flujo de pantallas o por los razonamientos que aconsejan LAZY.

(12) vs (9)+(38): Para llegar a la pantalla del caso de uso de (12), *Ver reserva*, primero tendremos que haber pasado por la pantalla del caso de uso de (9) o de (38), *Listar reservas persona/Listar reservas grupo*. En este caso, al disparar los casos de uso de *Listar reservas persona* o *Listar reservas grupo*, ya estamos cargando los datos de las reservas en memoria. Como previamente hemos disparado una listar, los datos ya están cargados en memoria por lo que una carga EAGER no es beneficiosa. Renunciamos a la carga EAGER.

(17) vs (12): Para llegar a la pantalla del caso de uso de (17), *Modificar reserva*, antes debemos pasar por la pantalla del caso de uso de (12), *Ver reserva*. Al igual que en el caso anterior, *Ver reserva*, no necesita una carga EAGER porque previamente ya se han cargado los datos por el caso de uso *Listar reservas persona/grupo*. Renunciamos a la carga EAGER.

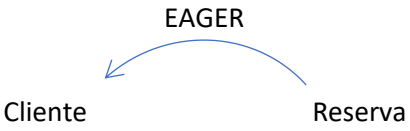
(39): La pantalla del caso de uso de (39) es una pantalla a la que no se llega a través de ninguna otra, por lo que todos los datos que aparezcan se han de cargar en memoria. En este caso, ya que queremos mostrar todas las reservas de la base de datos necesitamos cagarlas todas en memoria para poder mostrarlas en la pantalla. Necesitamos una carga EAGER.

Finalmente elegimos aplicar una carga **EAGER**.

Análisis de encadenamiento de cargas

Analizamos si, con la configuración que obtenemos al final, existen posibles encadenamientos de cargas EAGER que provoquen una carga excesiva de objetos en memoria

En nuestro caso, solo tenemos una carga EAGER desde Reserva a Cliente. No es posible que se produzca un encadenamiento de ningún tipo



Relación de métodos implementados

Operación	Clase + Método
Métodos para carga de propiedad LAZY	ClienteDao.recuperaCodigoCli(Cliente)
Consulta HQL con INNER JOIN	-
Consulta HQL con OUTER JOIN	-
Consulta HQL con subconsulta	-
Consulta HQL con función de agregación	-