|  |  |  |  |
| --- | --- | --- | --- |
| 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.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 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 () | LAZY |  |
| 2 (Editar persona) | Persona única | LAZY | LAZY |  |
| 3 (Eliminar persona) | Persona única  Colección de sus reservas | LAZY | LAZY | LAZY |
| 4 (Listar reservas persona) | Persona única: C  Colección de sus reservas |  | EAGER () | LAZY () |
| 4.1 (Ver reserva) | Persona o grupo únicos  Colección de todas las reservas | LAZY | LAZY () | EAGER () |
| 4.2 (Alta reserva) | Grupo o persona únicas | LAZY () | LAZY () |  |
| 4.3 (Modificar reserva) | Persona o grupo únicos  Reserva asociada | LAZY () | LAZY () | EAGER () |
| 4.4 (Eliminar reserva) | Persona o grupo únicos  Reserva asociada | LAZY () | EAGER () | LAZY () |
| 5 (Alta grupo) | - |  |  |  |
| 5.1 (Añadir integrantes)  6.1.2.1 (Añadir integrante) | Grupo único  Colección de integrantes | LAZY () | LAZY () |  |
| 6 (Listar grupos) | Todos los grupos: C1, C2, … | LAZY () | LAZY () |  |
| 6.1 (Ver grupo) | Grupo único | LAZY () | LAZY () |  |
| 6.1.2 (Listar integrantes) | Grupo único  Colección de todos sus integrantes | EAGER () | LAZY () |  |
| 6.1.2.2 (Eliminar integrante) | Grupo único  Colección de sus integrantes | LAZY () | LAZY () |  |
| 6.2 (Editar grupo) | Grupo único  Colección de sus integrantes | EAGER (19) | LAZY (20) |  |
| 6.3 (Eliminar grupo) | Todos los grupos | LAZY (21) | LAZY (22) | LAZY |
| 6.4 (Listar reservas grupo) | Grupo único  Colección de sus reservas | LAZY () | EAGER () | LAZY () |
| 7 (Listar reservas) | Todas las reservas |  |  | EAGER () |
| 7.1 (Filtro reservas por fecha) | Todas las reservas que cumplan la condición |  |  | LAZY () |
| 7.2 (Filtro reservas por nombre albergue) | Todas las reservas que cumplan la condición |  |  | LAZY () |
| ELECCIÓN FINAL |  |  |  |  |