Sistemas transaccionales – ISIS2304 – Entrega 1
Leidy J. Lozano Flórez
Laura M. Restrepo Palomino
Karen T. Vera Hernández

Documentación de los Requerimientos Funcionales

Requerimiento funcional 1

Nombre	RF1 - Registrar / actualizar / borrar / consultar tipos de usuarios
Resumen	Se implementan las operaciones CRUD para la relación tipos de usuarios. Usa los métodos del repositorio Tipo_usuario: - darTiposUsuario() - darTipoUsuario(@Param("id_tipo") long id_tipo) - eliminarTipoUsuario(@Param("id_tipo") long id_tipo) - actualizarTipoUsuario(@Param("id_tipo") long id_tipo, @Param("nombre") String nombre) - insertarTipoUsuario(@Param("nombre") String nombre)
Usuario	Administrador de la base de datos del hotel.
Entradas	
- [REGISTRAR] No recibe nada - [ACTUALIZAR] Recibe un ID del tipo de usuario que se desea actualizar junto con un atributo del modelo de tipo de usuario (Este incluye el nombre del tipo de usuario) que contiene la nueva información del tipo. - [ELIMINAR] Recibe el ID del tipo de usuario que se va a eliminar. - [CONSULTAR] Recibe el ID del tipo de usuario que se va a consultar.	
Resultados	
- Se registra un nuevo tipo de usuario en la base de datos del hotel, el cual puede ser sometido a las demás operaciones CRUD. - Se actualiza el nombre de un tipo de usuario en la base de datos del hotel. - Se elimina un tipo de usuario permanentemente de la base del hotel. - Se obtiene el nombre de tipo de usuario correspondiente al ID ingresado.	
RNF asociados	
RNF1 – Privacidad (Solo el administrador puede realizar estas operaciones) y RNF2 – Persistencia	

Requerimiento funcional 2

Nombre	RF2 - Registrar / actualizar / borrar / consultar usuario
Resumen	Se implementan las operaciones CRUD para la relación usuarios. Usa los métodos del repositorio usuario: - darUsuarios() - darUsuario(@Param("num_documento") String num_documento) - eliminarUsuario(@Param("num_documento") String num_documento) - actualizarUsuario(@Param("num_documento") String num_documento, @Param("num_documento_actualizado") String num_documento_actualizado, @Param("tipo_documento") String tipo_documento, @Param("nombre") String nombre, @Param("correo") String correo, @Param("tipo") Integer tipo, @Param("hotel") String hotel)

	- insertarUsuario(@Param("num_documento") String num_documento, @Param("tipo_documento") String tipo_documento, @Param("nombre") String nombre, @Param("correo") String correo, @Param("tipo") Integer tipo, @Param("hotel") String hotel)
Usuario	Administrador de la base de datos del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe el número de identificación usuario que se desea actualizar junto con un atributo del modelo de usuario (Este incluye Número de documento del usuario, tipo de documento del usuario, nombre del usuario, correo del usuario, ID numérico correspondiente a su tipo de usuario y NIT del hotel) que contiene la nueva información del tipo. - [ELIMINAR] Recibe el número de documento del usuario que se va a eliminar. - [CONSULTAR] Recibe el número de documento del usuario que se va a consultar.	
Resultados	
- Se registra un nuevo usuario en la base de datos del hotel, el cual puede ser sometido a las demás operaciones CRUD. - Se actualiza la información de un usuario en la base de datos del hotel. - Se elimina un usuario permanentemente de la base del hotel. - Se obtiene la información vinculada a un usuario dado su número de identificación.	
RNF asociados	
RNF1 – Privacidad (Solo el administrador puede realizar estas operaciones) y RNF2 – Persistencia	

Requerimiento funcional 3

Nombre	RF3 - Registrar / actualizar / borrar / consultar tipo de habitación
Resumen	Se implementan las operaciones CRUD para la relación tipos de habitación. Usa los métodos del repositorio Tipo_habitacion: - darTiposHabitacion() - darTipoHabitacion(@Param("id_tipo") long id_tipo) - eliminarTipoHabitacion(@Param("id_tipo") long id_tipo) - actualizarTipoHabitacion(@Param("id_tipo") long id_tipo, @Param("nombre") String nombre, @Param("dotacion") String dotacion, @Param("capacidad") Integer capacidad) - insertarTipoHabitacion(@Param("nombre") String nombre, @Param("dotacion") String dotacion, @Param("capacidad") Integer capacidad)
Usuario	Administrador de la base de datos del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe el ID de del tipo de habitación que se desea actualizar junto con un atributo del modelo de tipo de habitación (Este incluye ID del tipo de habitación, su nombre, su dotación y su capacidad) que contiene la nueva información del tipo. - [ELIMINAR] Recibe el número de documento del usuario que se va a eliminar. - [CONSULTAR] Recibe el número de documento del usuario que se va a consultar.	

Resultados
- Se registra un nuevo tipo de habitación en la base de datos del hotel, el cual puede ser sometido a las demás operaciones CRUD. - Se actualiza la información de un tipo de habitación en la base de datos del hotel. - Se elimina un tipo de habitación permanentemente de la base del hotel. - Se obtiene la información del tipo de habitación vinculado al ID ingresado.
RNF asociados
RNF1 – Privacidad (Solo el administrador puede realizar estas operaciones) y RNF2 – Persistencia

Requerimiento funcional 4

Nombre	RF4 - registrar / actualizar / borrar / consultar habitación
Resumen	Se implementan las operaciones CRUD para la relación de habitaciones. Usa los métodos del repositorio habitación: - darHabitaciones() - darHabitacion(@Param("num_habitacion") Integer num_habitacion) - eliminarHabitacion(@Param("num_habitacion") Integer num_habitacion) - actualizarHabitacion(@Param("num_habitacion") Integer num_habitacion, @Param("num_habitacion_actualizado") Integer num_habitacion_actualizado, @Param("costo_noche") Double costo_noche, @Param("tipo") Integer tipo, @Param("hotel") String hotel) - insertarHabitacion(@Param("num_habitacion") Integer num_habitacion, @Param("costo_noche") Double costo_noche, @Param("tipo") Integer tipo, @Param("hotel") String hotel)
Usuario	Administrador de la base de datos del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe el número de la habitación que se desea actualizar junto con un atributo del modelo de habitación (Este incluye su número de habitación, el costo que tiene por noche, un identificador del tipo de habitación y el NIT del hotel al cual pertenece) que contiene la nueva información de la habitación. - [ELIMINAR] Recibe el número de habitación de la habitación que se va a eliminar. - [CONSULTAR] Recibe el número de documento de la habitación que se va a consultar.	
Resultados	
- Se registra una nueva habitación en la base de datos del hotel, la cual puede ser sometida a las demás operaciones CRUD. - Se actualiza la información de una habitación en la base de datos del hotel. - Se elimina una habitación permanentemente de la base del hotel. - Se obtiene la información de una habitación vinculado a cierto número de habitación.	
RNF asociados	
RNF1 – Privacidad (Solo el administrador puede realizar estas operaciones) y RNF2 – Persistencia	

Requisito funcional 5

Nombre	RF5 - Registrar / actualizar / borrar / consultar un servicio del hotel
Resumen	Se implementan las operaciones CRUD para las relaciones que corresponden a los servicios del hotel. Usa los métodos del repositorio servicio: - darServicios() - darServicio(@Param("tipo") Integer tipo) - eliminarServicio(@Param("tipo") Integer tipo) - actualizarServicio(@Param("tipo") Integer tipo, @Param("tipo_actualizado") Integer tipo_actualizado, @Param("capacidad") Integer capacidad, @Param("tipo_cobro") String tipo_cobro) - insertarServicio(@Param("tipo") Integer tipo, @Param("capacidad") Integer capacidad, @Param("tipo_cobro") String tipo_cobro)
Usuario	Administrador de la base de datos del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe un entero que identifica al servicio junto con un atributo del modelo de servicio (Este incluye el ID del servicio, la capacidad (para los servicios que aplican) y el tipo de cobro que aplica a ese servicio) que contiene la nueva información del servicio. - [ELIMINAR] Recibe el ID del servicio que se va a eliminar. - [CONSULTAR] Recibe el ID del servicio que se va a consultar.	
Resultados	
- Se registra un nuevo servicio en la base de datos del hotel, la cual puede ser sometida a las demás operaciones CRUD. - Se actualiza la información de un servicio en la base de datos del hotel. - Se elimina un servicio permanentemente de la base del hotel dado un ID de servicio. - Se obtiene la información de un servicio vinculado a cierto ID de servicio.	
RNF asociados	
RNF1 – Privacidad (Solo el administrador puede realizar estas operaciones) y RNF2 – Persistencia	

Requisito funcional 6

Nombre	RF6 - Registrar / actualizar / borrar / consultar un plan de consumo
Resumen	Se implementan las operaciones CRUD para la relación que corresponden a los planes de consumo del hotel. Usa los métodos del repositorio plan_Consumo: - darPlanesConsumo() - darPlanConsumo(@Param("tipo") Integer tipo) - eliminarPlanConsumo(@Param("tipo") Integer tipo) - actualizarPlanConsumo(@Param("tipo") Integer tipo, @Param("tipo_actualizado") Integer tipo_actualizado, @Param("genera_descuento") String genera_descuento, @Param("tiempo_estadia") Integer tiempo_estadia, @Param("descuento_ser_com") String descuento_ser_com, @Param("valor_descuento_ser") Double valor_descuento_ser)

	- insertarPlanConsumo(@Param("tipo") Integer tipo, @Param("genera_descuento") String genera_descuento, @Param("tiempo_estadia") Integer tiempo_estadia, @Param("descuento_ser_com") String descuento_ser_com, @Param("valor_descuento_ser") Double valor_descuento_ser)
Usuario	Administrador de la base de datos del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe un entero que identifica al plan de consumo junto con un atributo del modelo de plan de consumo (Este incluye el ID del tipo de plan del consumo, un booleano que indica si genera descuento sobre el costo del alojamiento, el tiempo de estadía mínimo que involucra el plan, un booleano que aclara si genera descuento en lo servicios comerciales y el valor del descuento sobre este servicio (Si es que aplica)) que contiene la nueva información del plan. - [ELIMINAR] Recibe el ID del plan de consumo que se va a eliminar. - [CONSULTAR] Recibe el ID del plan de consumo que se va a consultar.	
Resultados	
- Se registra un nuevo plan de consumo en la base de datos del hotel, la cual puede ser sometida a las demás operaciones CRUD. - Se actualiza la información de un plan de consumo en la base de datos del hotel. - Se elimina un plan de consumo permanentemente de la base del hotel dado un ID de plan de consumo. - Se obtiene la información de un plan de consumo vinculado a cierto ID de plan de consumo.	
RNF asociados	
RNF1 – Privacidad (Solo el administrador puede realizar estas operaciones) y RNF2 – Persistencia	

Requisito funcional 7

Nombre	RF7 - Registrar / actualizar / borrar / consultar una reserva de alojamiento
Resumen	Se implementan las operaciones CRUD para la relación que corresponden a las reservas de alojamiento del hotel. Usa los métodos del repositorio reserva: - darReservas() - darReserva(@Param("codigo_reserva") Integer codigo_reserva) - eliminarReserva(@Param("codigo_reserva") Integer codigo_reserva) - actualizarReserva(@Param("codigo_reserva") Integer codigo_reserva, @Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida, @Param("num_huespedes") Integer num_huespedes, @Param("entro") String entro, @Param("salio") String salio, @Param("paz_y_salvo") String paz_y_salvo, @Param("cliente_reserva") String cliente_reserva,

	@Param("hotel") String hotel, @Param("habitacion") Integer habitacion, @Param("plan_consumo") Integer plan_consumo) - insertarReserva(@Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida, @Param("num_huespedes") Integer num_huespedes, @Param("entro") String entro, @Param("salio") String salio, @Param("paz_y_salvo") String paz_y_salvo, @Param("cliente_reserva") String cliente_reserva, @Param("hotel") String hotel, @Param("habitacion") Integer habitacion, @Param("plan_consumo") Integer plan_consumo) - darReservasHabitacionFechas(@Param("num_habitacion") Integer num_habitacion, @Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida)
Usuario	Cliente del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe un código numérico que identifica la reserva junto con un atributo del modelo de reserva(Este incluye el código de la reserva, la fecha de inicio de la reserva, la fecha de finalización, el número de huéspedes incluidos en la reserva, un booleano que indica si los huéspedes ya están en el hotel, un booleano que indica si los huéspedes ya salieron, un booleano que indica si el cliente está a paz y salvo con sus pagos, el número de identificación del huésped que realizó la reserva, el NIT del hotel, el número de habitación se reservó y el ID del plan de consumo adquirido) que contiene la nueva información de la reserva. - [ELIMINAR] Recibe el código de la reserva que se va a eliminar. - [CONSULTAR] Recibe e código de la reserva que se va a consultar.	
Resultados	
- Se registra una nueva reserva en la base de datos del hotel, la cual puede ser sometida a las demás operaciones CRUD. - Se actualiza la información de una reserva en la base de datos del hotel. - Se elimina una reserva plan de la base del hotel dado un código de reserva. - Se obtiene la información de una reserva vinculado a a cierto código de reserva.	
RNF asociados	
RNF1 – Privacidad (Solo el cliente puede realizar estas operaciones) y RNF2 – Persistencia	

Requisito funcional 8

Nombre	RF8 - registrar / actualizar / borrar / consultar una reserva de un servicio del hotel por parte de un cliente
Resumen	Se implementan las operaciones CRUD para la relación que corresponden a las reservas de servicios al interior del hotel. Usa los métodos del repositorio reserva servicio: - darReservasServicio() - darReservaServicio(@Param("id_reserva_servicio") Integer id_reserva_servicio)

	- eliminarReservaServicio(@Param("id_reserva_servicio") Integer id_reserva_servicio) - actualizarReservaServicio(@Param("id_reserva_servicio") Integer id_reserva_servicio, @Param("dia_reserva") Date dia_reserva, @Param("hora_inicio") Timestamp hora_inicio, @Param("hora_fin") Timestamp hora_fin, @Param("servicio_comercial") Integer servicio_comercial, @Param("servicio_renta") Integer servicio_renta) - insertarReservaServicio(@Param("dia_reserva") Date dia_reserva, @Param("hora_inicio") Timestamp hora_inicio, @Param("hora_fin") Timestamp hora_fin, @Param("servicio_comercial") Integer servicio_comercial, @Param("servicio_renta") Integer servicio_renta)
Usuario	Cliente del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe ID numérico que identifica la reserva del servicio junto con un atributo del modelo de reserva de servicio (Este incluye el ID de la reserva del servicio, el día que se reservó, la hora de inicio de la reserva, la hora de finalización de la reserva y el ID de un servicio comercial o el ID de un servicio de renta) que contiene la nueva información de la reserva del servicio. - [ELIMINAR] Recibe el ID numérico que identifica la reserva del servicio que se va a eliminar. - [CONSULTAR] Recibe el ID numérico que identifica la reserva del servicio que se va a consultar.	
Resultados	
- Se registra una nueva reserva de servicio en la base de datos del hotel, la cual puede ser sometida a las demás operaciones CRUD. - Se actualiza la información de una reserva de servicio ya existente en la base de datos del hotel. - Se elimina una reserva de servicio plan de la base del hotel dado un ID numérico. - Se obtiene la información de una reserva de servicio vinculado a ID numérico de reserva de servicio.	
RNF asociados	
RNF1 – Privacidad (Solo el cliente puede realizar estas operaciones) y RNF2 – Persistencia	

Requisito funcional 9

Nombre	RF9 - Registrar / actualizar / borrar / consultar la llegada de un cliente al hotel
Resumen	Se implementan las operaciones CRUD para el atributo de la relación reserva asociado a la entrada de un cliente al hotel. Usa los métodos del repositorio reserva: - darReservas() - darReserva(@Param("codigo_reserva") Integer codigo_reserva) - eliminarReserva(@Param("codigo_reserva") Integer codigo_reserva)

	- actualizarReserva(@Param("codigo_reserva") Integer codigo_reserva, @Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida, @Param("num_huespedes") Integer num_huespedes, @Param("entro") String entro, @Param("salio") String salio, @Param("paz_y_salvo") String paz_y_salvo, @Param("cliente_reserva") String cliente_reserva, @Param("hotel") String hotel, @Param("habitacion") Integer habitacion, @Param("plan_consumo") Integer plan_consumo) - insertarReserva(@Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida, @Param("num_huespedes") Integer num_huespedes, @Param("entro") String entro, @Param("salio") String salio, @Param("paz_y_salvo") String paz_y_salvo, @Param("cliente_reserva") String cliente_reserva, @Param("hotel") String hotel, @Param("habitacion") Integer habitacion, @Param("plan_consumo") Integer plan_consumo) darReservasHabitacionFechas(@Param("num_habitacion") Integer num_habitacion, @Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida)
Usuario	Recepcionista del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe el código que identifica a una reserva junto con un atributo del modelo de reserva de naturaleza reserva (Este incluye el código de la reserva, la fecha de inicio de la reserva, la fecha de finalización, el número de huéspedes incluidos en la reserva, un booleano que indica si los huéspedes ya están en el hotel, un booleano que indica si los huéspedes ya salieron, un booleano que indica si el cliente está a paz y salvo con sus pagos, el número de identificación del huésped que realizó la reserva, el NIT del hotel, el número de habitación se reservó y el ID del plan de consumo adquirido) que contiene la nueva información de la reserva del servicio, entre esta información, está la información pertinente a la entrada del huésped. - [ELIMINAR] Recibe el código que identifica la reserva que contiene la información de entrada que se desea eliminar. - [CONSULTAR] Recibe el código que identifica la reserva que contiene la información de entrada que se desea consultar.	
Resultados	
- Se registra una nueva reserva de servicio en la base de datos del hotel, la cual podrá ser actualizada para incluir información de la entrada del cliente. - Se actualiza la información de una reserva de alojamiento dado un código y una nueva información, esta información puede actualizar la información de entrada del cliente. - Se elimina la reserva de alojamiento con cierto código que contenía la información sobre la entrada del cliente. - Se obtiene la información de una reserva asociada a cierto código que tiene entre sus atributos la información de entrada del cliente	
RNF asociados	
RNF1 – Privacidad (Solo el recepcionista puede realizar estas operaciones) y RNF2 – Persistencia	

Leidy J. Lozano Flórez
Laura M. Restrepo Palomino
Karen T. Vera Hernández

Requisito funcional 10

Nombre	RF10 - Registrar / actualizar / borrar / consultar un consumo de un servicio del hotel por parte de un cliente o sus acompañantes
Resumen	Se implementan las operaciones CRUD para la relación que corresponden a los servicios consumidos por los clientes. Usa los métodos del repositorio reserva_Servicio: - darConsumen() - darConsume(@Param("cliente") String cliente, @Param("servicio") long servicio, @Param("fecha_hora") Timestamp fecha_hora) - eliminarConsume(@Param("cliente") String cliente, @Param("servicio") long servicio, @Param("fecha_hora") Timestamp fecha_hora) - actualizarConsume(@Param("cliente") String cliente, @Param("servicio") long servicio, @Param("fecha_hora") Timestamp fecha_hora, @Param("cliente_actualizado") String cliente_actulizado, @Param("servicio_actualizado") long servicio_actualizado, @Param("fecha_hora_actualizado") Timestamp fecha_hora_actualizado, @Param("cuenta_pagada") String cuenta_pagada) - insertarConsume(@Param("cliente") String cliente, @Param("servicio") long servicio, @Param("fecha_hora") Timestamp fecha_hora, @Param("cuenta_pagada") String cuenta_pagada)
Usuario	Empleado del hotel.
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe el número de identificación del cliente que consumió el servicio, el identificador numérico del servicio y la hora en la que se consumió. Estos tres identifican el consumo del servicio, adicionalmente, se recibe un atributo del modelo de naturaleza consumo de servicio (Incluye el número de identificación del cliente que consumió el servicio, el identificador numérico del servicio, la hora y un booleano que indica si la cuenta fue pagada) que contiene la nueva información del consumo del servicio. - [ELIMINAR] Recibe el número de identificación del cliente que consumió el servicio, el identificador numérico del servicio y la hora en la que se consumió, lo que identifica el consumo del servicio que se va a eliminar. - [CONSULTAR] Recibe el número de identificación del cliente que consumió el servicio, el identificador numérico del servicio y la hora en la que se consumió, lo que identifica el consumo de servicio que se va a consultar.	
Resultados	
- Se registra un nuevo consumo de servicio en la base de datos del hotel, la cual puede ser sometida a las demás operaciones CRUD. - Se actualiza la información de un consumo de servicio ya existente en la base de datos del hotel.	

- Se elimina un consumo de servicio plan de la base del hotel dado un número de identificación del cliente que consumió el servicio, el identificador numérico del servicio y la hora en la que se consumió. - Se obtiene la información de consumo de servicio vinculado a un número de identificación del cliente que consumió el servicio, el identificador numérico del servicio y la hora en la que se consumió el servicio.
RNF asociados
RNF1 – Privacidad (Solo el empleado puede realizar estas operaciones) y RNF2 – Persistencia

Requisito funcional 11

Nombre	RF11 - registrar / actualizar / borrar / consultar la salida de un cliente
Resumen	Se implementan las operaciones CRUD para el atributo de la relación reserva asociado a la salida de un cliente al hotel. Usa los métodos del repositorio reserva: - darReservas() - darReserva(@Param("codigo_reserva") Integer codigo_reserva) - eliminarReserva(@Param("codigo_reserva") Integer codigo_reserva) - actualizarReserva(@Param("codigo_reserva") Integer codigo_reserva, @Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida, @Param("num_huespedes") Integer num_huespedes, @Param("entro") String entro, @Param("salio") String salio, @Param("paz_y_salvo") String paz_y_salvo, @Param("cliente_reserva") String cliente_reserva, @Param("hotel") String hotel, @Param("habitacion") Integer habitacion, @Param("plan_consumo") Integer plan_consumo) - insertarReserva(@Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida, @Param("num_huespedes") Integer num_huespedes, @Param("entro") String entro, @Param("salio") String salio, @Param("paz_y_salvo") String paz_y_salvo, @Param("cliente_reserva") String cliente_reserva, @Param("hotel") String hotel, @Param("habitacion") Integer habitacion, @Param("plan_consumo") Integer plan_consumo) darReservasHabitacionFechas(@Param("num_habitacion") Integer num_habitacion, @Param("fecha_entrada") Date fecha_entrada, @Param("fecha_salida") Date fecha_salida)
Usuario	Recepcionista del hotel
Entradas	
- [REGISTRAR] No recibe nada. - [ACTUALIZAR] Recibe el código que identifica a una reserva junto con un atributo del modelo de reserva de naturaleza reserva (Este incluye el código de la reserva, la fecha de inicio de la reserva, la fecha de finalización, el número de huéspedes incluidos en la reserva, un booleano que indica si los huéspedes ya están en el hotel, un booleano que indica si los huéspedes ya salieron, un booleano que indica si el cliente está a paz y salvo con sus pagos, el número de identificación del huésped que realizó la reserva, el NIT del	

hotel, el número de habitación se reservó y el ID del plan de consumo adquirido) que contiene la nueva información de la reserva del servicio, entre esta información, está la información pertinente a la salida del huésped. - [ELIMINAR] Recibe el código que identifica la reserva que contiene la información de salida que se desea eliminar. - [CONSULTAR] Recibe el código que identifica la reserva que contiene la información de salida que se desea consultar.
Resultados
- Se registra una nueva reserva de servicio en la base de datos del hotel, la cual podrá ser actualizada para incluir información de la salida del cliente. - Se actualiza la información de una reserva de alojamiento dado un código y una nueva información, esta información puede actualizar la información de salida del cliente. - Se elimina la reserva de alojamiento con cierto código que contenía la información sobre la salida del cliente. - Se obtiene la información de una reserva asociada a cierto código que tiene entre sus atributos la información de salida del cliente
RNF asociados
RNF1 – Privacidad (Solo el recepcionista puede realizar estas operaciones) y RNF2 – Persistencia