Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Ingeniería del Software

CURSO 2025-2026

Memoria

Universidad
Zaragoza
1542

Universidad Zaragoza

ÍNDICE

Introducción y objetivos

El principal objetivo es realizar el análisis inicial de un sistema de gestión de reservas de quads, que sirva como base para fases posteriores de diseño, implementación y pruebas.

En esta práctica se aborda el proceso de especificación de requisitos, tanto funcionales como no funcionales, así como la definición de las restricciones del sistema. A partir de ellos, se han elaborado los casos de uso y sus flujos de eventos, representados mediante diagramas UML (casos de uso, clases y secuencia) que describen la estructura y el comportamiento dinámico del sistema.

Los objetivos principales de este trabajo son:

- Identificar y priorizar los requisitos del sistema (con MoSCoW).
- Establecer las restricciones que limitan el funcionamiento del sistema.
- Modelar los casos de uso y detallar sus flujos principales y alternativos.
- Representar gráficamente la estructura estática del sistema mediante un diagrama de clases.
- Describir el comportamiento dinámico del sistema mediante diagramas de secuencia.
- Sentar las bases para la posterior fase de diseño e implementación de la aplicación de gestión de reservas de quads.

Requisitos

Nº Requisito	Requisito funcional	MoSCoW
R.F. 1	El Sistema debe permitir la adición de un nuevo quad, almacenando su matrícula, tipo de quad, precio en euros del alquiler por día y descripción.	Must
R.F. 2	El Sistema debe permitir la consulta de un listado de quads previamente creados, pudiéndose ordenar por matrícula, tipo o precio.	Must
R.F. 3	El Sistema debe permitir la modificación de los quads previamente creados, así como la actualización del precio de las reservas a las que esté asociado dicho quad.	Must
R.F 4	El Sistema debe permitir la eliminación de los quads, así	Must

Nº Requisito	Requisito funcional	MoSCoW
	como la actualización del precio de las reservas a las que esté asociado dicho quad. previamente creados.	
R.F. 5	El Sistema debe permitir la creación de una reserva, indicando un nombre de cliente, el número móvil del cliente, la fecha de recogida y devolución, y una selección de quads, indicando también si se va a necesitar casco para su conducción.	Must
R.F. 6	El Sistema debería permitir la consulta de un listado de las reservas previamente creadas, pudiéndose ordenar por nombre de cliente, número de móvil, fecha de recogida, o fecha de devolución.	Must
R.F. 7	El Sistema debe permitir la modificación de las reservas previamente creadas.	Must
R.F. 8	El Sistema debe permitir la eliminación de las reservas previamente creadas.	Must
R.F. 9	El Sistema debe permitir calcular automáticamente el precio total de una reserva	Must
R.F. 10	El Sistema debe permitir el envío al móvil del cliente de la información de la reserva (incluido precio)	Must

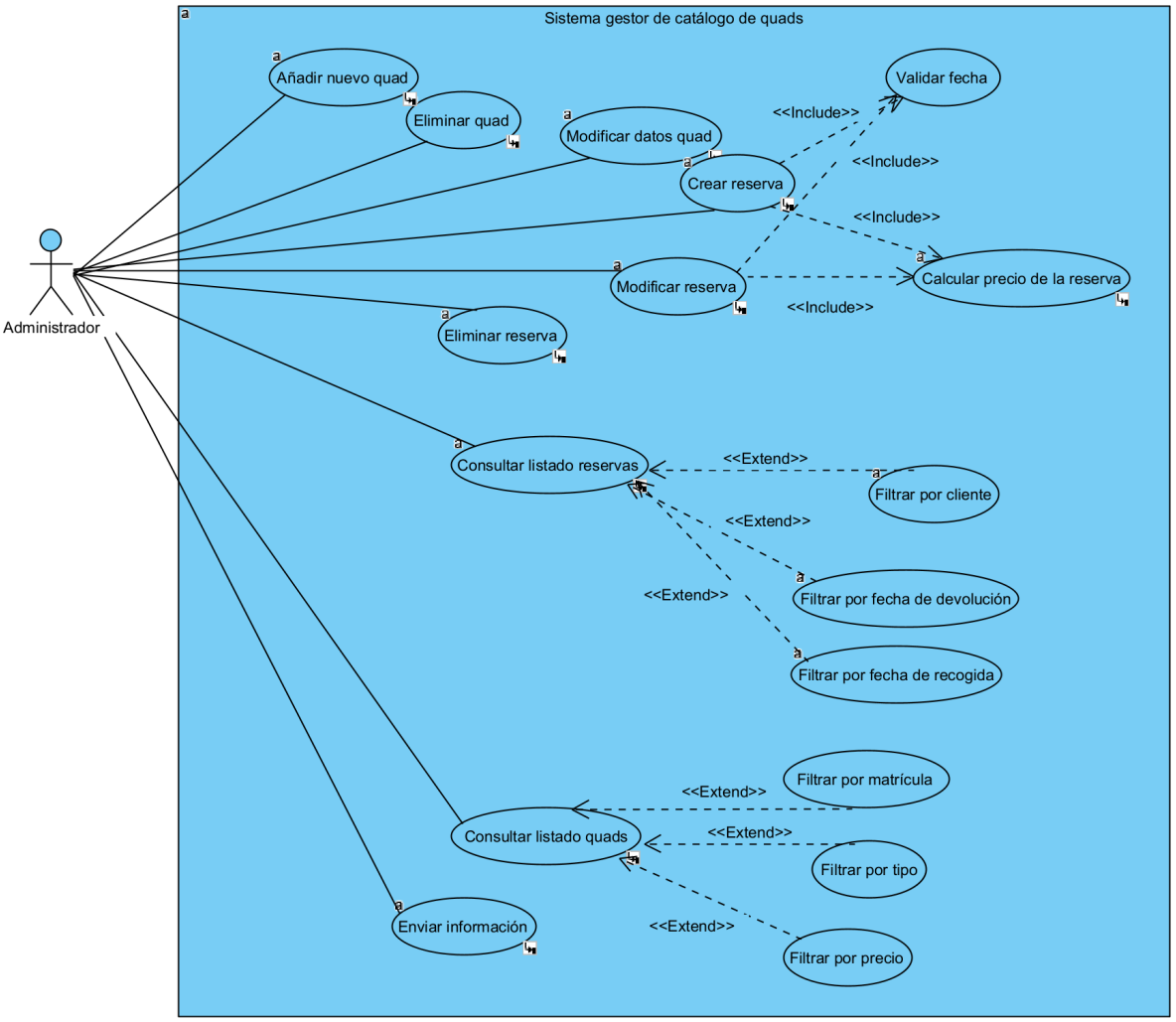
Nº Requisito	Requisito no funcional	MoSCoW
R.N.F. 1	El Sistema no permitirá la creación de un nuevo quad si hay 100 quads existentes.	
R.N.F. 2	El Sistema no permitirá la creación de una nueva reserva si hay 20000 reservas existentes.	

Restricciones

La aplicación debe estar disponible para los dispositivos con un sistema operativo Android.

Análisis

Diagrama de casos de uso



Añadir nuevo quad:

Flujo principal:

1. El caso de uso comienza cuando el administrador selecciona la opción “Añadir quad” desde la pantalla principal.
2. El sistema muestra un formulario solicitando los datos del quad: matrícula, tipo, precio por día y descripción.
3. El administrador rellena los campos y pulsa el botón “CONFIRMAR”.
4. El sistema valida los datos, guarda el quad en la base de datos y muestra un mensaje de confirmación.
5. El caso de uso termina volviendo a la pantalla principal.

Flujo alternativo:

- En el paso 3, el administrador pulsa “Aceptar” sin rellenar todos los campos requeridos o con datos inválidos, como una matrícula duplicada.
- El sistema muestra un mensaje de error indicando los campos a corregir.
- El caso de uso continúa en el paso 2 del flujo principal.

Consultar listado de quads

Flujo principal:

1. El administrador selecciona la opción “Consultar listado de quads”.
2. El sistema muestra un listado de los quads disponibles.
3. extend Ordenar listado de quads por matrícula, tipo o precio.
4. El caso de uso termina cuando el administrador regresa a la pantalla principal.

Flujo alternativo:

- En el paso 2, si no existen quads registrados, el sistema muestra un mensaje de advertencia “No hay quads registrados”.
- El caso de uso finaliza al cerrar el aviso.

Modificar datos quad

Flujo principal:

1. El caso de uso parte del flujo de “Consultar listado de quads”.
2. El administrador selecciona un quad del listado y pulsa “Modificar”.
3. El sistema muestra un formulario con los datos actuales del quad.
4. El administrador modifica los campos deseados y pulsa “Aplicar cambios”.
5. El sistema actualiza los datos y muestra un mensaje de confirmación.
6. El listado se actualiza con la información modificada.

Flujo alternativo:

- En el paso 4, el administrador deja algún campo requerido vacío o introduce un valor inválido.
- El sistema muestra los campos en rojo y un mensaje indicando el error.
- El caso de uso continúa en el paso 3.

Eliminar quad

Flujo principal:

1. El caso de uso parte de “Consultar listado de quads”.
2. El administrador selecciona un quad y pulsa la opción “Eliminar”.
3. El sistema muestra un mensaje de confirmación.
4. El administrador pulsa “Aceptar” y el quad se elimina del sistema.
5. El caso de uso termina actualizando el listado.

Flujo alternativo:

- En el paso 4, el administrador pulsa “Cancelar”.
- El quad no se elimina y el caso de uso termina sin cambios.

Crear reserva

Flujo principal:

1. El administrador selecciona la opción “Crear reserva”.
2. El sistema muestra un formulario solicitando nombre del cliente, número de móvil, fechas de recogida y devolución, selección de quad y cascos necesarios.
3. El administrador rellena los datos y pulsa “Aceptar”.
4. include Validar fechas de la reserva.
5. include Calcular precio total automáticamente.
6. El sistema guarda la reserva y muestra un mensaje de confirmación.

Flujo alternativo:

- En el paso 3, algún campo requerido no está relleno y el sistema muestra un mensaje de error.
- En el paso 4, las fechas no son válidas, por ejemplo devolución antes que recogida, y el sistema muestra error y devuelve al paso 2.

Consultar listado de reservas

Flujo principal:

1. El administrador selecciona la opción “Consultar reservas”.
2. El sistema muestra un listado de las reservas existentes.
3. extend Ordenar listado de reservas por cliente, móvil, fecha recogida o devolución.
4. El caso de uso termina cuando el administrador vuelve a la pantalla principal.

Flujo alternativo:

- En el paso 2, si no existen reservas registradas, el sistema muestra un mensaje de advertencia “No hay reservas registradas”.
- El caso de uso finaliza al cerrar el aviso.

Modificar reserva

Flujo principal:

1. El caso de uso parte de “Consultar listado de reservas”.
2. El administrador selecciona una reserva y pulsa “Modificar”.
3. El sistema muestra un formulario con los datos actuales de la reserva.
4. El administrador edita los campos y pulsa “Aplicar cambios”.
5. include Validar fechas de la reserva.
6. include Recalcular precio total.
7. El sistema actualiza los datos y muestra un mensaje de confirmación.

Flujo alternativo:

- En el paso 4, algún campo requerido queda vacío o inválido y el sistema marca los errores y vuelve al paso 3.
- En el paso 5, las fechas no son válidas y el sistema muestra error y se vuelve al paso 3.

Eliminar reserva

Flujo principal:

1. El caso de uso parte de “Consultar listado de reservas”.
2. El administrador selecciona una reserva y pulsa “Eliminar”.
3. El sistema solicita confirmación.
4. El administrador pulsa “Aceptar” y la reserva se elimina.
5. El listado se actualiza y el caso de uso termina.

Flujo alternativo:

- En el paso 4, el administrador pulsa “Cancelar”.
- La reserva no se elimina y el caso de uso termina sin cambios.

Calcular precio de la reserva

Flujo principal:

1. El caso de uso comienza cuando el sistema recibe los datos de una reserva (fechas y quad seleccionado).
2. El sistema calcula la duración de la reserva en días.
3. El sistema multiplica el número de días por el precio por día del quad seleccionado.
4. Si se han solicitado cascos, añade el suplemento correspondiente.
5. El sistema guarda el precio total en la reserva y lo muestra en pantalla.
6. El caso de uso termina devolviendo el control al caso de uso padre.

Flujo alternativo:

- En el paso 2, si las fechas no son válidas (ej. devolución anterior a recogida), el sistema no puede calcular el precio y muestra un mensaje de error.
- El caso de uso retorna al paso 2 del flujo principal de “Crear reserva” o “Modificar reserva”.

Nota: Si el administrador modifica el precio diario de un quad, las reservas ya existentes no se ven afectadas, conservando el precio total calculado en el momento de su creación. Solo las nuevas reservas usarán el precio actualizado.

Enviar información de la reserva al cliente

Flujo principal:

1. El caso de uso comienza cuando al menos una reserva ha sido validada y guardada en el sistema.

   Nota: Una reserva se considera válida cuando:

   - Las fechas de recogida y devolución son coherentes (la devolución es posterior a la recogida).
   - El quad seleccionado está disponible en ese rango de fechas.
   - Se ha proporcionado un nombre y número de teléfono válidos del cliente.
   - Se ha calculado correctamente el precio total asociado a la reserva.
2. El administrador selecciona una reserva y le da al botón de “Enviar Información”
3. El sistema prepara un mensaje con los datos de la reserva: nombre del cliente, fechas, quad seleccionado, cascos y precio total.

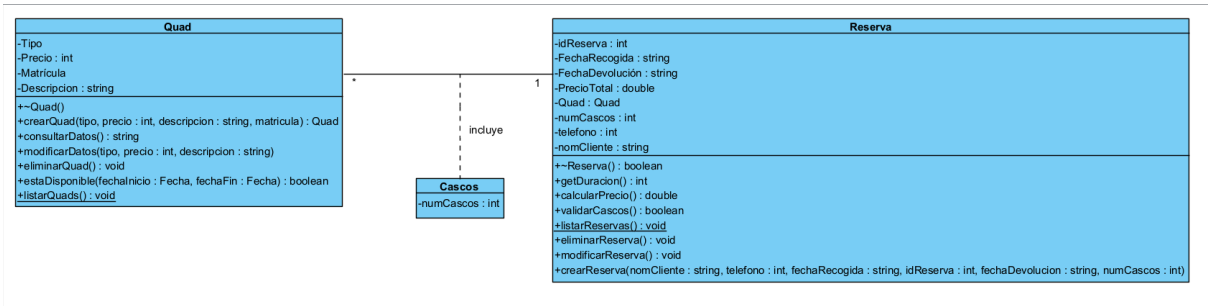
4. El sistema envía el mensaje al número de móvil proporcionado por el cliente.
5. El caso de uso termina devolviendo el control al caso de uso padre.

Flujo alternativo:

- En el paso 3, si no se puede enviar el mensaje (ej. número inválido), el sistema muestra un aviso al administrador.
- El caso de uso termina sin enviar la notificación, pero la reserva queda registrada.

Diagrama de clases

El diagrama de clases presentado a continuación representa el modelo estático del sistema de gestión de alquiler de quads a nivel de análisis. Este modelo captura las entidades fundamentales del dominio del problema, sus atributos, operaciones y relaciones, sin considerar aún aspectos de implementación o interfaz de usuario.



Clase Quad

Representa los vehículos disponibles para alquiler en el sistema.

Atributos

- **Tipo**: Clasificación del quad (monoplaza o biplaza)
- **Precio** (*double*): Coste de alquiler por día en euros
- **Matrícula** (*string*): Identificador único del vehículo (formato: 4 dígitos + 3 letras)
- **Descripción** (*string*): Información adicional sobre marca, modelo, color, etc.

Operaciones

- **Quad(matricula: String, tipo: String, precioDia: double, descripcion: String):** Constructor para crear nuevos quads.
- **~Quad()**: Destructor de la clase
- **consultarDatos(): string**
  Devuelve la información completa del quad.

- **modificarDatos(tipo, precio: int, descripción: string): void**
  Permite actualizar los datos del quad.
- **eliminarQuad(): void**
  Elimina el quad del sistema.
- **estaDisponible(fechaInicio, fechaFin): boolean**
  Verifica si el quad está disponible en un rango de fechas.
- **listarQuads(): void**
  Operación de clase para obtener todos los quads del sistema.

Clase Reserva

Representa las reservas realizadas por los clientes.

Atributos

- **idReserva** (*int*): Identificador único de la reserva
- **nomCliente** (*string*): Nombre del cliente.
- **telefono** (*int*): Número de teléfono del cliente
- **FechaRecogida** (*string*): Fecha de inicio del alquiler
- **FechaDevolución** (*string*): Fecha de fin del alquiler
- **PrecioTotal** (*double*): Coste total de la reserva

Operaciones

- **crearReserva(idReserva: int, nomCliente: String, telefono: int, fechaRecogida: String, fechaDevolucion: String, quad: Quad, numCascos: int):** Constructor para crear nuevas reservas.
- **~Reserva():** Destructor de reserva
- **getDuration(): int**
  Calcula la duración de la reserva en días.
- **calcularPrecio(): double**
  Determina el precio total basado en la duración y el quad.
- **validarFechas(): boolean**
  Verifica que las fechas sean coherentes.
- **validarCascos(): boolean**
  Comprueba que el número de cascos sea adecuado para el tipo de quad.
- **listarReservas(): void**
  Operación de clase para obtener todas las reservas.
- **enviarInformacion(): void**
  Envía al cliente los datos de la reserva (nombre, fechas, quad, cascos y precio total) al número de teléfono registrado.

Entidad Cascos

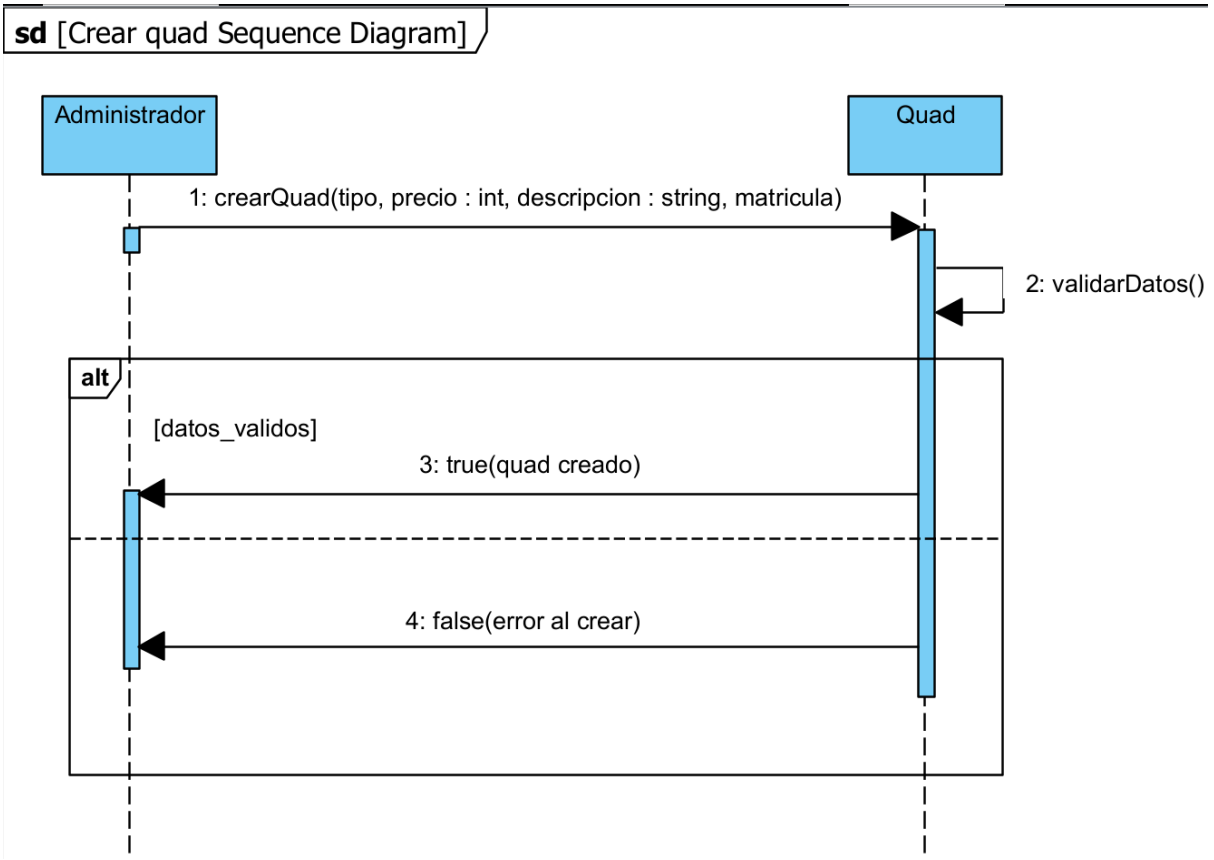
Aunque en el diagrama inicial aparece como clase separada, se considera que numCascos debería ser un atributo de la relación Quad_Reserva, ya que los cascos no tienen existencia independiente en el sistema.

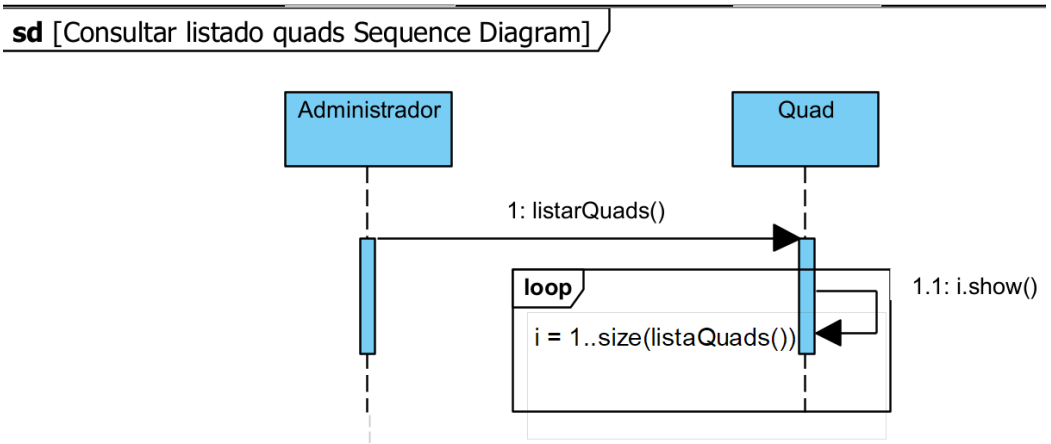
Relaciones entre clases

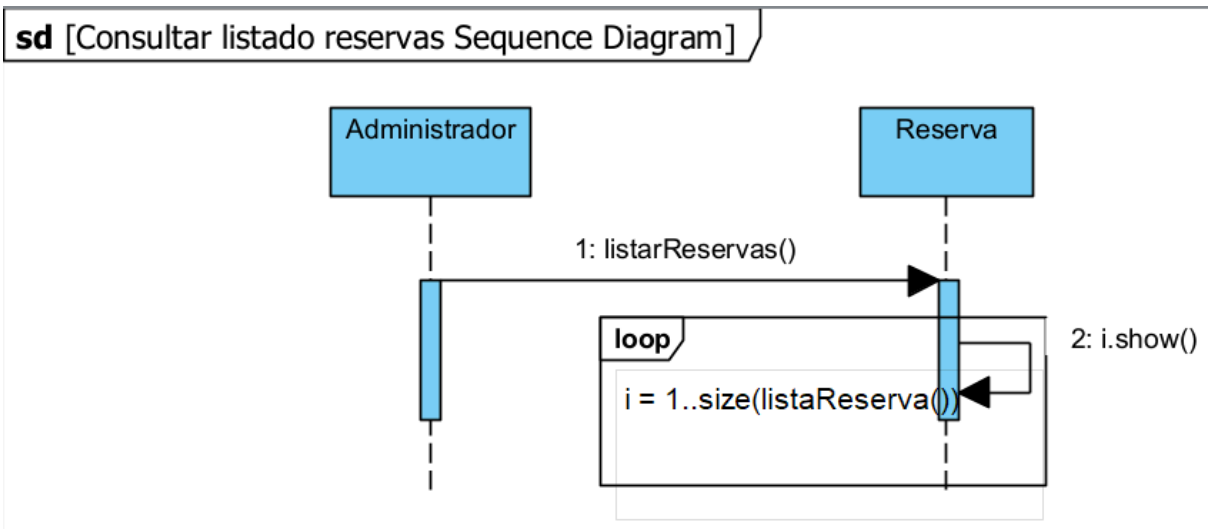
Reserva & Quad: Relación de asociación donde cada reserva hace referencia a los quad alquilados.

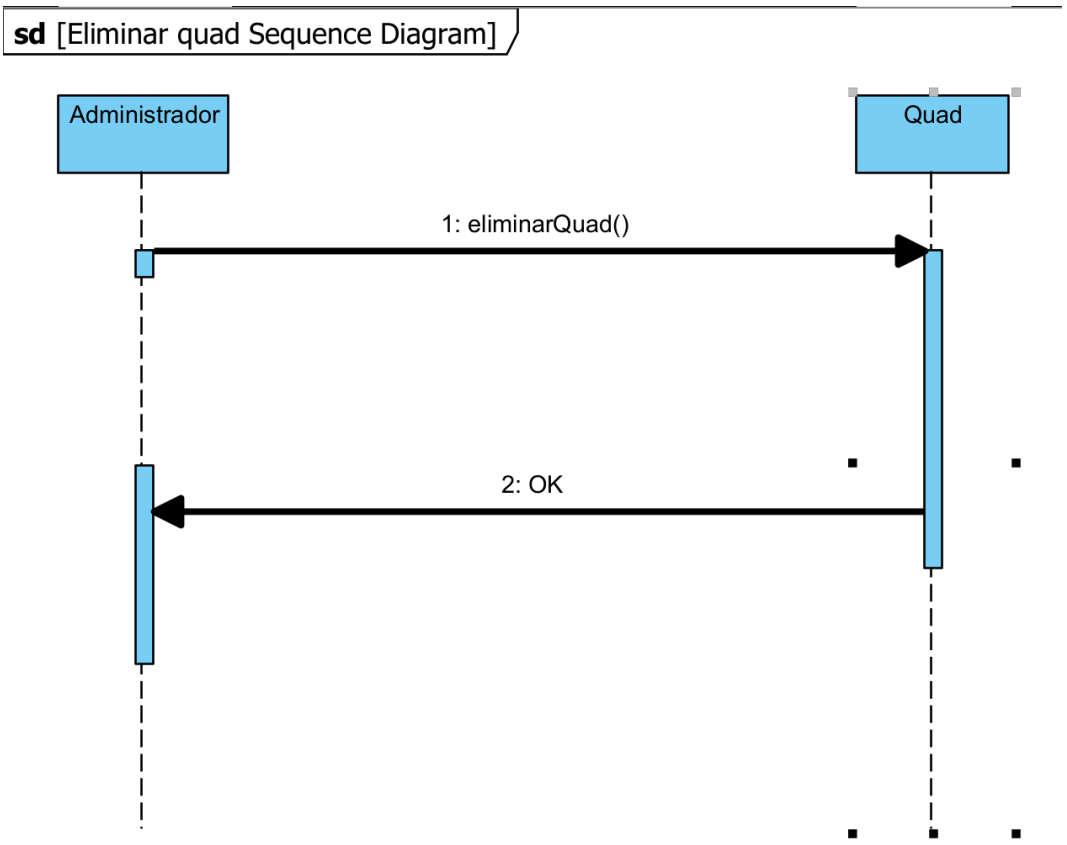
Multiplicidad: Una reserva está asociada a uno o varios quads y un quad puede tener de cero a múltiples reservas (N:M).

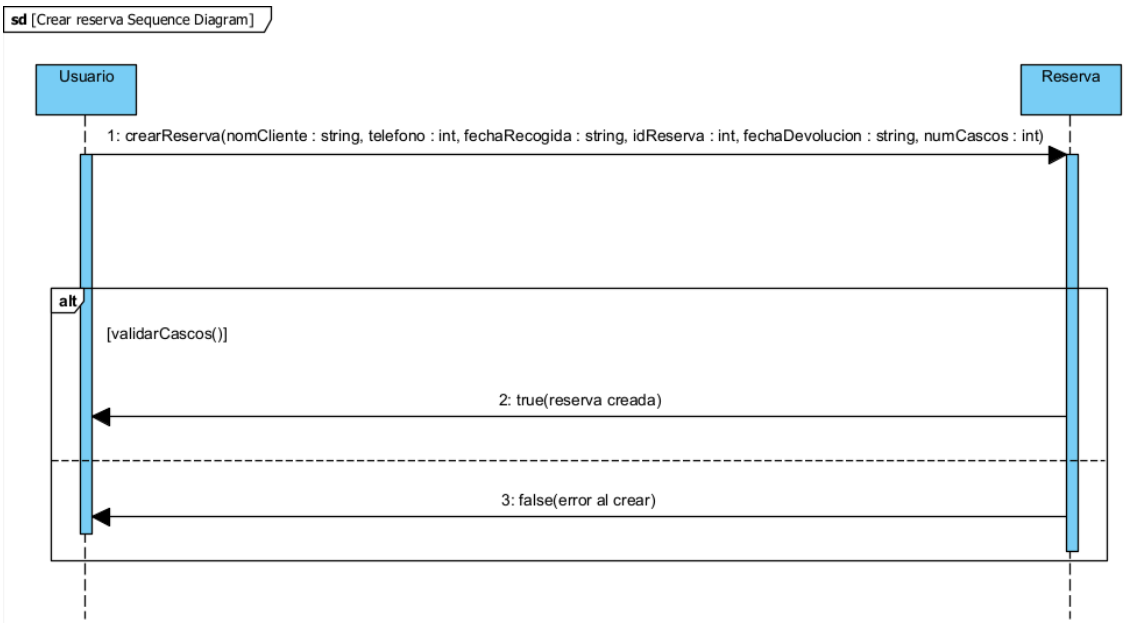
Diagramas de secuencia

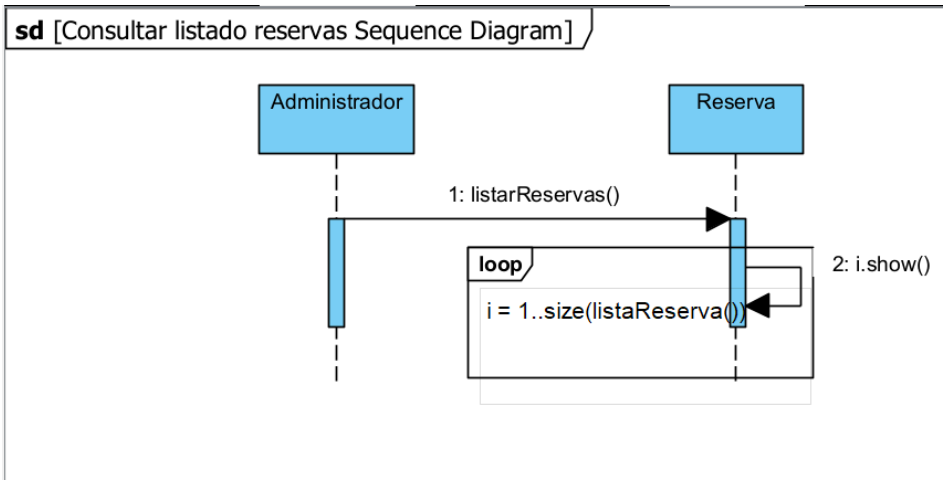


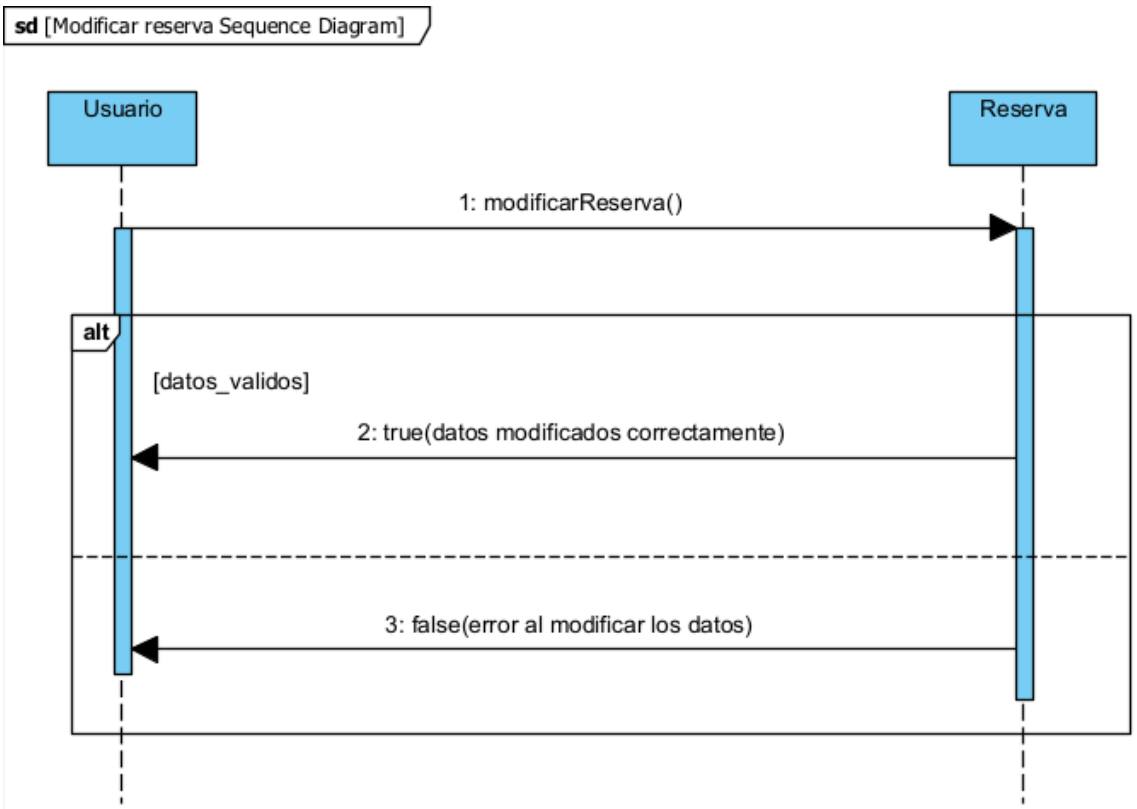


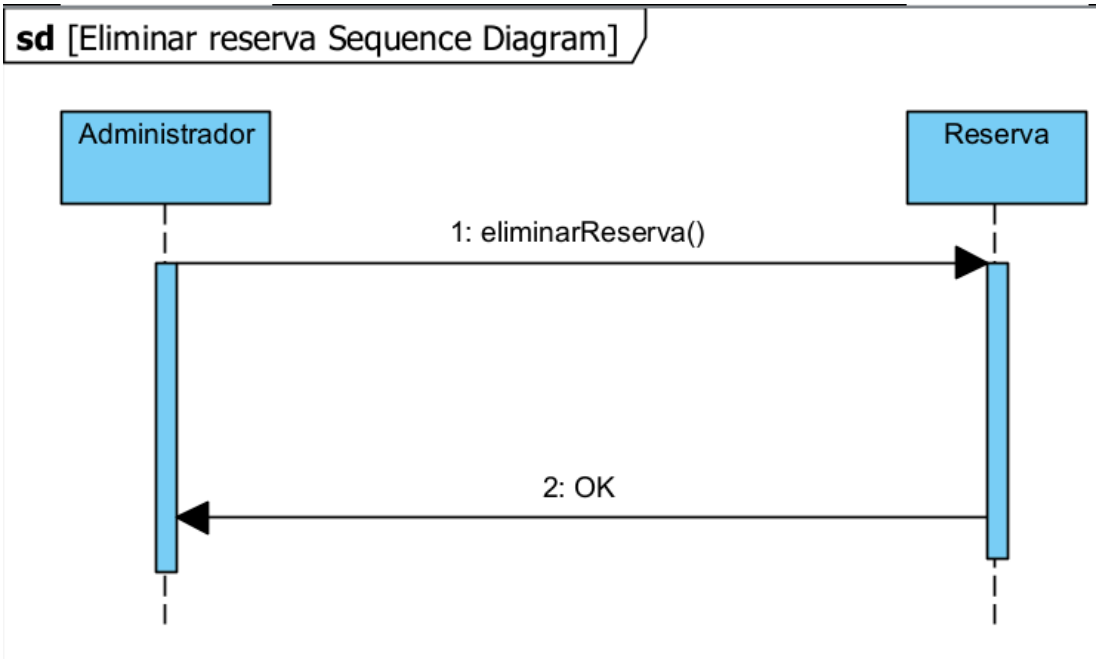

sd [Consultar listado reservas Sequence Diagram]
Administrador
Reserva
1: listarReservas()
loop
i = 1..size(listaReserva())
2: i.show()


sd [Eliminar quad Sequence Diagram]
Administrador
Quad
1: eliminarQuad()
2: OK

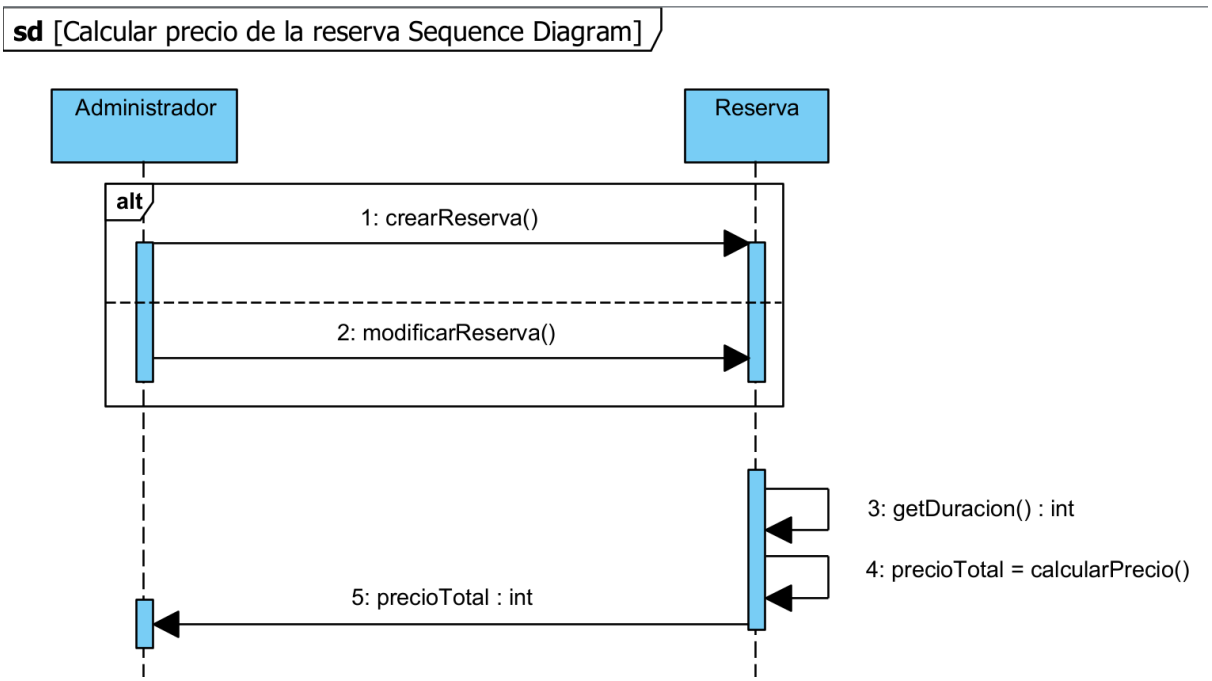

sd [Crear reserva Sequence Diagram]
Usuario
Reserva
1: crearReserva(nomCliente : string, telefono : int, fechaRecogida : string, idReserva : int, fechaDevolucion : string, numCascos : int)
alt
[validarCascos()]
2: true(reserva creada)
3: false(error al crear)

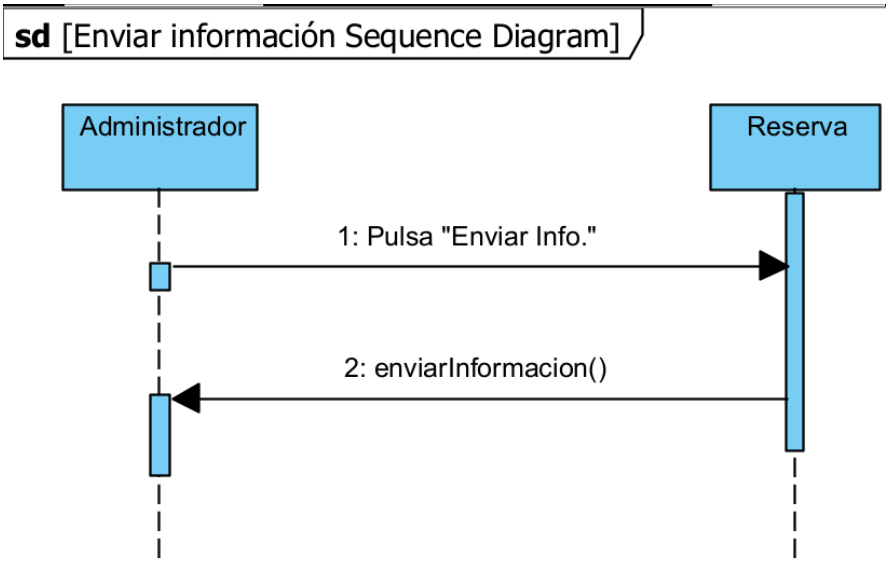

sd [Consultar listado reservas Sequence Diagram]
Administrador
Reserva
1: listarReservas()
loop
i = 1..size(listaReserva())
2: i.show()


sd [Modificar reserva Sequence Diagram]
Usuario
Reserva
1: modificarReserva()
alt
[datos_validos]
2: true(datos modificados correctamente)
3: false(error al modificar los datos)


sd [Eliminar reserva Sequence Diagram]
Administrador
Reserva
1: eliminarReserva()
2: OK

-

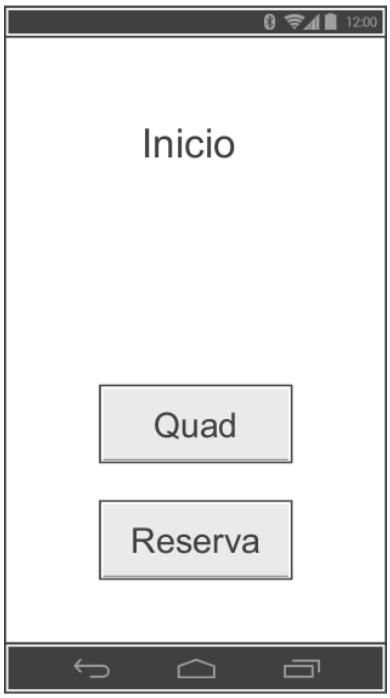

sd [Calcular precio de la reserva Sequence Diagram]
Administrador
Reserva
alt
1: crearReserva()
2: modificarReserva()
3: getDuracion() : int
4: precioTotal = calcularPrecio()
5: precioTotal : int


sd [Enviar información Sequence Diagram]
Administrador
Reserva
1: Pulsa "Enviar Info."
2: enviarInformacion()

Prototipos de pantalla

El prototipado de la interfaz para el usuario se ha hecho a partir de Visual Paradigm, usando la herramienta *Android Phone Wireframe*. El diseño de la interfaz sigue estrictamente las especificaciones definidas en los requisitos funcionales definidos anteriormente, pudiéndose realizar todos los casos de uso también definidos.

Las pantallas se muestran a continuación:

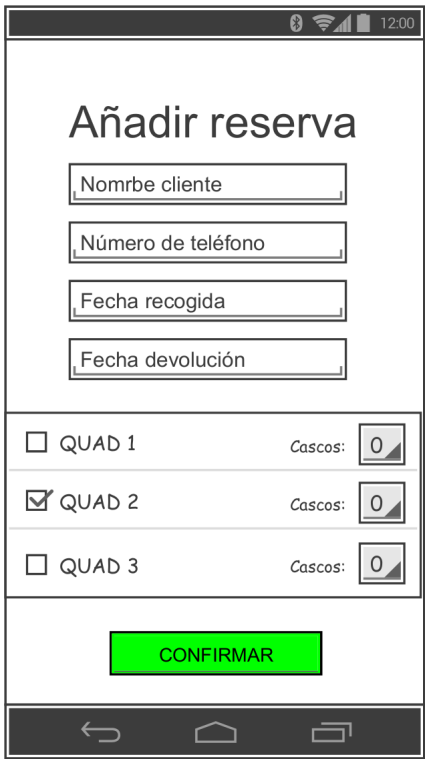



12:00
Lista de quads
Filtrar por:
Matrícula
Tipo
Precio
QUAD 1
QUAD 2
QUAD 3


12:00
Quad
Datos
Eliminar

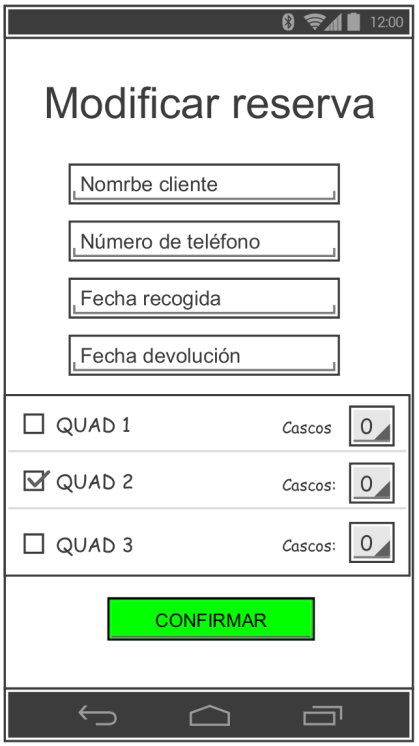

12:00
Modificar quad
Matrícula
Precio/Día(€)
Monoplaza
Biplaza
Descripción
CONFIRMAR


12:00
Reserva
Añadir reserva
Ver reservas


12:00
Añadir reserva
Nomrbe cliente
Número de teléfono
Fecha recogida
Fecha devolución
QUAD 1
Cascos:
0
QUAD 2
Cascos:
0
QUAD 3
Cascos:
0
CONFIRMAR


12:00
Lista de reservas
Filtrar por:
Cliente
Fecha recogida
Fecha devolución
RESERVA 1
RESERVA 2
RESERVA 3


12:00
Reserv
(Precio
Enviar info.
Datos
Eliminar


12:00
Modificar reserva
Nomrbe cliente
Número de teléfono
Fecha recogida
Fecha devolución
QUAD 1
Cascos
0
QUAD 2
Cascos:
0
QUAD 3
Cascos:
0
CONFIRMAR

La interfaz del sistema se ha diseñado para ser sencilla, intuitiva y funcional. Desde la pantalla de inicio, el administrador puede acceder a los apartados “Quads” y “Reservas”. En el apartado Quads, el sistema permite añadir un nuevo quad o consultar el listado de quads registrados. En dicho listado, los quads pueden filtrarse y ordenarse por matrícula, tipo o precio. Al seleccionar un quad concreto, se muestra una pantalla con sus datos, desde la cual es posible modificar su información o eliminarlo.

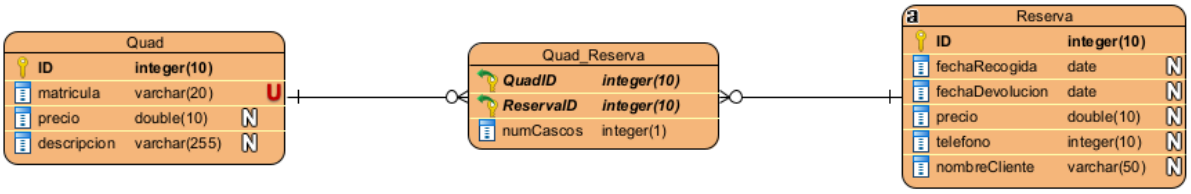
El apartado Reservas funciona de forma análoga: se pueden crear nuevas reservas, consultar el listado existente, modificar o eliminar una reserva. Además, desde la pantalla de detalle de cada reserva, el sistema ofrece la opción de enviar la información de la reserva al cliente.

No se incluyen pantallas de retroalimentación así como manejo de errores o feedback para éxito por simplicidad en la fase de desarrollo de diseño. Estas se incluirán en la fase del desarrollo del prototipo funcional.

Mapa de navegación

Modelo lógico de la base de datos relacional

El desarrollo de este modelo se ha llevado a cabo mediante la herramienta Entity Relationship Diagram del paquete Database Modeling. El diagrama consta de 2 entidades principales, Quad y Reserva, unidas mediante una relación N:M de la cual sale la entidad Quad_Reserva, dentro de la cual se ha añadido el atributo numCascos, que representa el número de cascos asignados a uno de los quads en una reserva determinada.
El diagrama se muestra a continuación:




```
-- -----------------------------------------------------
-- Tabla: Quad
-- Basada en la configuración que acabamos de hacer.
-- -----------------------------------------------------
CREATE TABLE Quad (
  id_quad INTEGER PRIMARY KEY AUTOINCREMENT,
  matricula VARCHAR(20) NOT NULL UNIQUE,
  descripcion TEXT
);

-- -----------------------------------------------------
-- Tabla: Reserva
-- -----------------------------------------------------
CREATE TABLE Reserva (
  id_reserva INTEGER PRIMARY KEY AUTOINCREMENT,
  nombre_cliente VARCHAR(255) NOT NULL,
  telefono_cliente VARCHAR(20),
  fecha_recogida DATETIME NOT NULL,
  fecha_devolucion DATETIME NOT NULL,
  estado_reserva VARCHAR(50) NOT NULL
);

-- -----------------------------------------------------
-- Tabla: Quad_Reserva (Tabla intermedia para N:M)
-- Resuelve la relación Muchos a Muchos entre Reserva y Quad.
-- -----------------------------------------------------
CREATE TABLE Quad_Reserva (
  id_reserva_fk INTEGER NOT NULL,
  id_quad_fk INTEGER NOT NULL,
  numCascos INTEGER NOT NULL,
```

```
  PRIMARY KEY (id_reserva_fk, id_quad_fk),
  FOREIGN KEY (id_reserva_fk) REFERENCES Reserva (id_reserva),
  FOREIGN KEY (id_quad_fk) REFERENCES Quad (id_quad)
);
```

Diagrama de paquetes

El paquete “ui” es el encargado de todos los aspectos relacionados con la interfaz del usuario. Este se descompone a su vez en los sub-paquetes “uiReserva” y “uiUsuario”, que agrupan la lógica de presentación (como ViewModels, Activities, etc.) para la gestión específica de las reservas y el manejo de los perfiles de usuario, respectivamente.

Por otro lado, el paquete “database” es el encargado de todos los aspectos relacionados con la base de datos del sistema (DAO, Repositorios, AppQuadsRoomDatabase). Como indican las flechas de dependencia, tanto “uiReserva” como “uiUsuario” necesitan acceder al paquete 'database' para consultar y persistir la información.

Finalmente, el paquete “send” es el encargado de todo lo relacionado con la gestión de una comunicación cómoda con el usuario (como el envío de notificaciones o correos de confirmación).

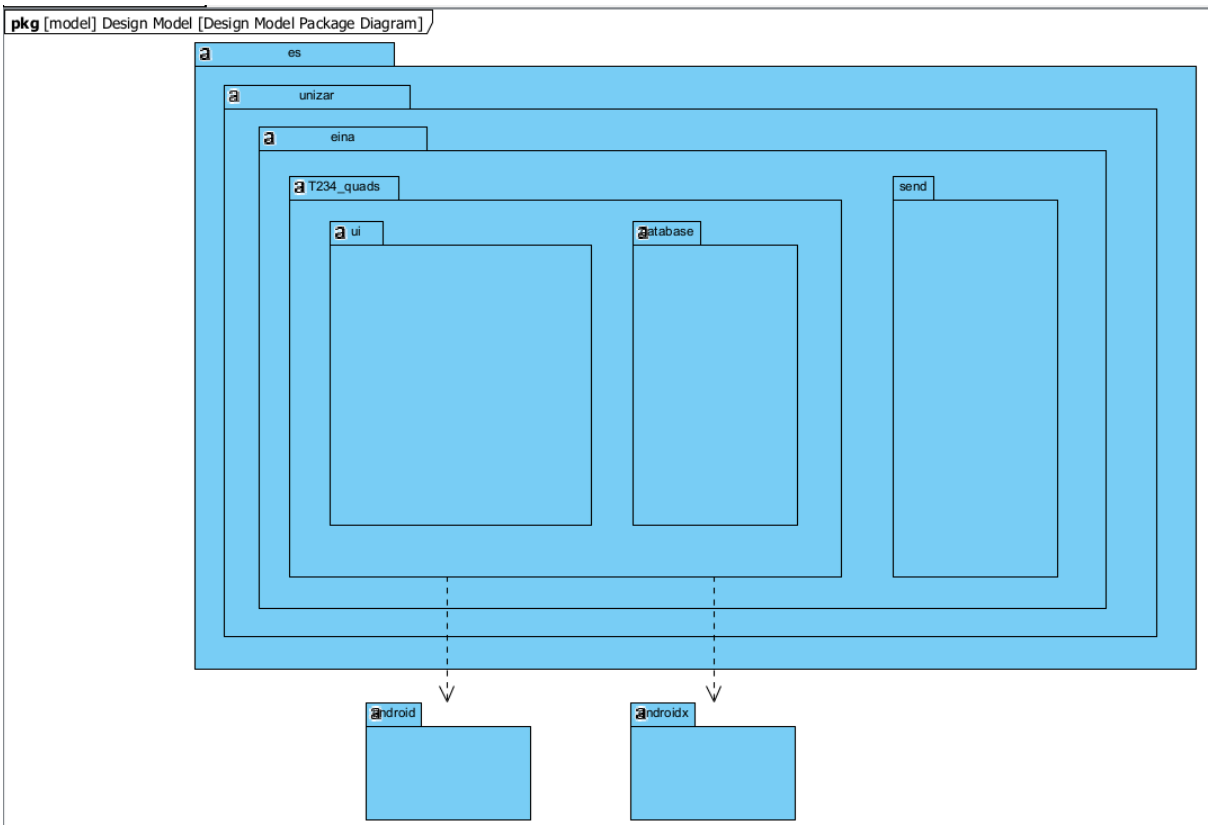


Diagrama de componentes

El componente “GestionReservasQuad” representa la aplicación de gestión de reservas de quads y el componente “Local Database” representa la base de datos que almacena los quads y las reservas de la aplicación.

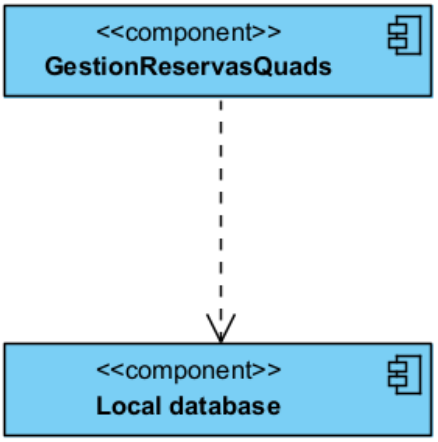
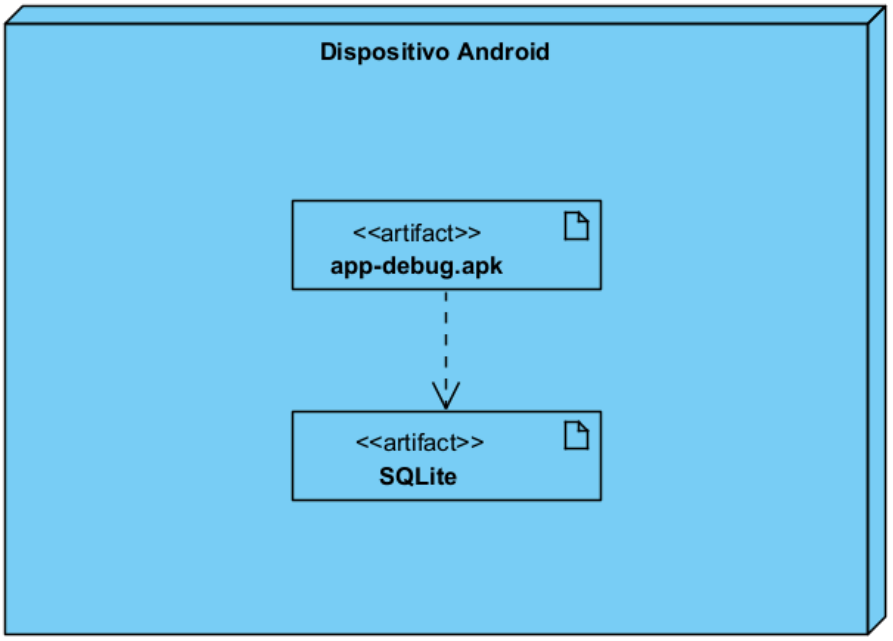


Diagrama de despliegue

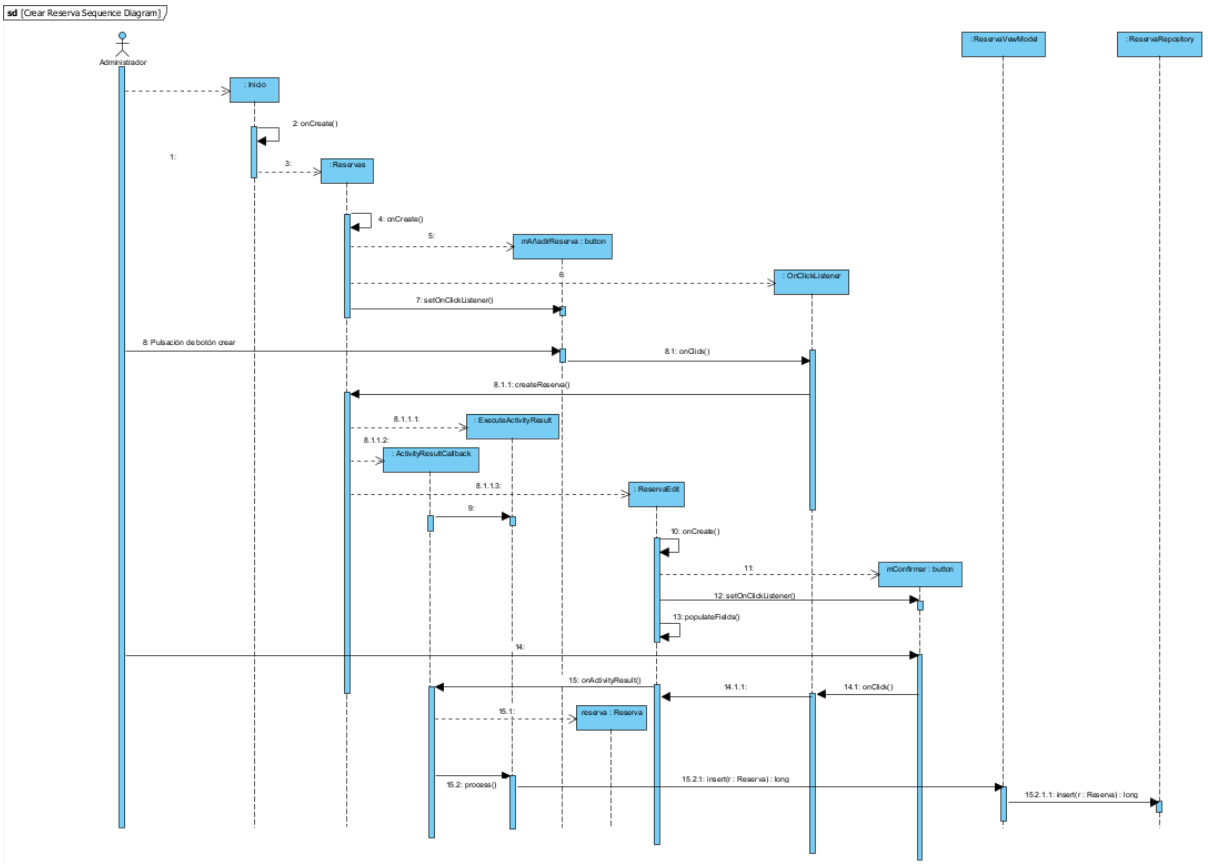


Hay 2 artefactos: 'app-debug.apk', que es el paquete instalable de la aplicación generado al compilar el proyecto, y 'SQLite', que representa la base de datos local que utiliza la aplicación para almacenar y gestionar los datos.

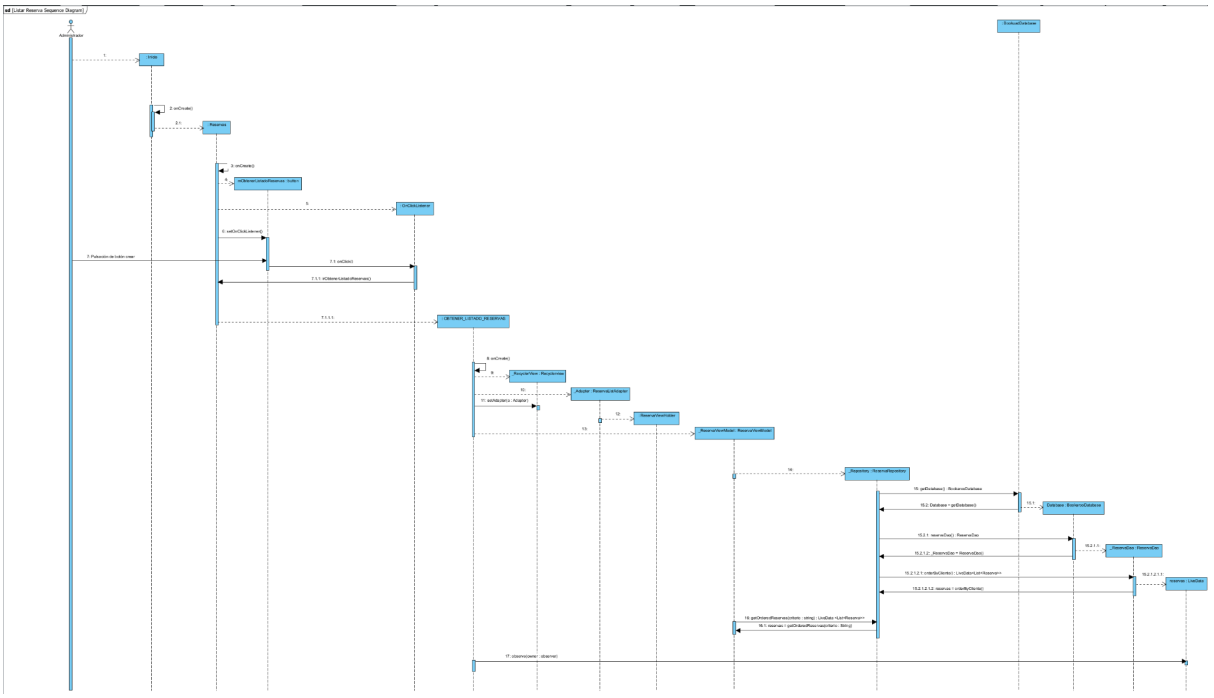
Diseño de objetos

Diagramas de secuencia

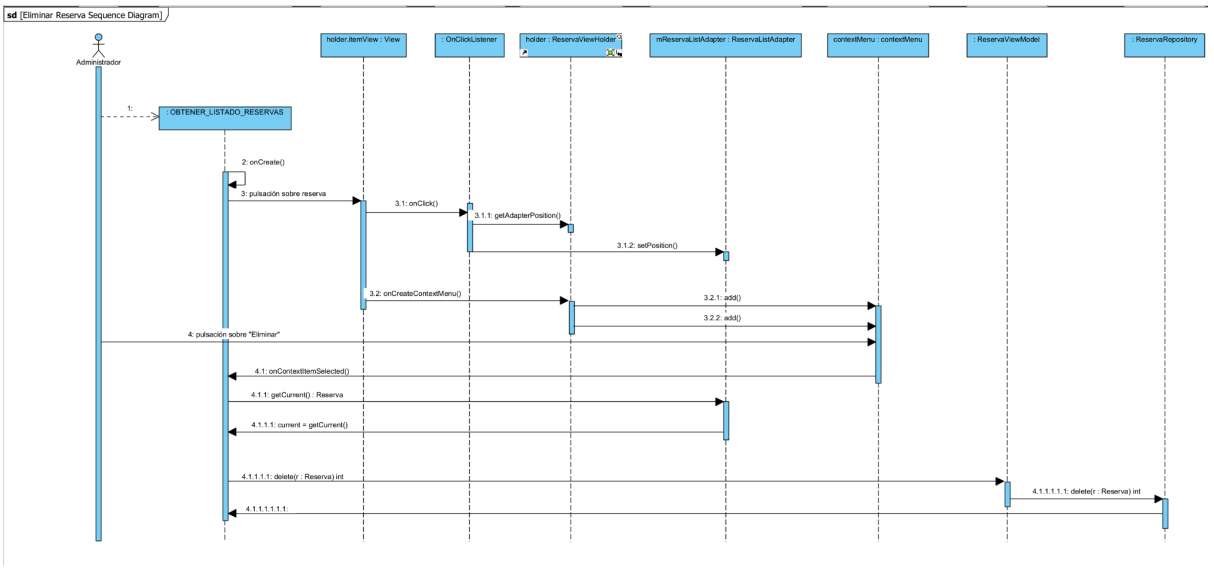
Crear reserva



Obtener listado de reservas



Eliminar Reserva



Bibliografía

Visual Paradigm Forums. Foro oficial de Visual Paradigm.
Disponible en: https://forums.visual-paradigm.com/ Discuss the Visual Paradigm

Stack Overflow. Sección “visual-paradigm” en Stack Overflow.
Disponible en: https://stackoverflow.com/questions/tagged/visual-paradigm