

INGENIERÍA SOFTWARE

PRÁCTICA 6

874055 - Ariana Porroche Llorén

871627 - Irene Pascual Albericio

1. RESUMEN

Hemos desarrollado la aplicación T242_Camping para el propietario de un camping. Esta aplicación almacena las parcelas disponibles, así como gestiona las distintas reservas de los clientes.

Por un lado, el propietario puede añadir las parcelas que tenga su camping a la base de datos, y consultar un listado de ellas. Así mismo, puede añadir las reservas que le van realizando los clientes, guardando sus datos personales, las fechas y las parcelas a reservar. También puede consultar un listado de todas las reservas. Estos listados se pueden filtrar según el criterio que le interese al propietario en cada momento.

Para el desarrollo de la aplicación hemos utilizado Visual Paradigm 17.2, una herramienta de modelado visual utilizada para diseñar y documentar sistemas de software. Con ella hemos diseñado los distintos diagramas (diagrama de casos de uso, diagrama de clases, diagramas de secuencia, diagrama de paquetes, diagrama de componentes y diagrama de despliegue).

Una vez diseñada la aplicación, comenzamos con la implementación. Para ello utilizamos Android Studio, un entorno de desarrollo integrado para el desarrollo de aplicaciones Android, creado por Google. Proporciona herramientas específicas para desarrollar, probar y depurar aplicaciones en dispositivos Android. Fuimos definiendo las diferentes clases y los distintos estilos para desarrollar la aplicación del camping diseñada previamente.

Tras todo ello, obtuvimos una aplicación funcional y adaptada a las necesidades del propietario, capaz de gestionar de manera eficiente tanto las parcelas como las reservas de los clientes. Gracias a un buen diseño y una buena implementación, logramos desarrollar una solución robusta, bien estructurada y fácil de usar, siempre mejorando la experiencia del usuario.

Por último, diseñamos distintas pruebas para comprobar la solidez y eficiencia de la aplicación, ofreciendo dicha opción en el menú de la aplicación. Estas pruebas nos ayudaron principalmente a nosotras, las desarrolladoras, a detectar ciertos fallos y poderlos corregir.

2. ÍNDICE

1. RESUMEN.....	1
2. ÍNDICE.....	2
3. INTRODUCCIÓN Y OBJETIVOS.....	3
4. REQUISITOS.....	4
4.1. CATÁLOGO DE REQUISITOS.....	4
4.2. DIAGRAMA DE CASOS DE USO.....	6
5. ANÁLISIS.....	12
5.1 DIAGRAMA DE CLASES.....	12
5.2 DIAGRAMAS DE SECUENCIA.....	12
6. PROTOTIPOS DE PANTALLAS.....	17
7. MAPA DE NAVEGACIÓN.....	25
8. DIAGRAMA DE PAQUETES.....	26
9. DIAGRAMA DE COMPONENTES.....	27
10. DIAGRAMA DE DESPLIEGUE.....	27
11. DIAGRAMA DE CLASES A NIVEL DE DISEÑO.....	28
12. DIAGRAMAS DE SECUENCIA A NIVEL DE DISEÑO.....	30
13. MODELO LÓGICO DE LA BASE DE DATOS RELACIONAL.....	35
14. DISEÑO DE LA IMPLEMENTACIÓN:.....	36
15. PRUEBAS.....	38
15.1. PRUEBAS DE CAJA NEGRA.....	38
15.2. PRUEBAS DE VOLUMEN.....	49
15.3. PRUEBAS DE SOBRECARGA.....	49
16. BIBLIOGRAFÍA.....	50

3. INTRODUCCIÓN Y OBJETIVOS

Primeramente, hicimos un análisis de los requisitos que debía cumplir nuestra aplicación. Para ello, capturamos los requisitos establecidos en la primera práctica de la asignatura y creamos el diagrama de casos de uso. A continuación, desarrollamos el diagrama de clases y los diagramas de secuencia de diseño, para hacernos una idea de la interacción entre los distintos componentes.

En este momento, desarrollamos un prototipo de las pantallas, centrándonos en la funcionalidad y las acciones que deberíamos implementar a posteriori. También el mapa de navegación nos ayudó a visualizar las interacciones entre las pantallas.

Una vez diseñados los aspectos esenciales, ya entramos en aspectos más relacionados con la implementación, como el diagrama de paquetes, el diagrama de despliegue o el diagrama de componentes, así como unos nuevos diagramas de secuencia centrados en la implementación. También diseñamos la base de datos, la cual almacenará las parcelas, las reservas y las parcelas asociadas a cada reserva.

Hecho ya el diseño, implementamos la aplicación en Android Studio, para lo que desarrollamos el código de las distintas clases establecidas previamente, y fuimos probando a medida que añadimos distintas funcionalidades.

Por último, para probar más exhaustivamente que cumplía con los requisitos establecidos, implementamos 3 tipos de pruebas: de caja negra, de volumen y de sobrecarga. Así ya tuvimos una aplicación robusta, con un uso fácil e intuitivo.

En cuanto a los objetivos, el principal objetivo que siempre tuvimos presente fue diseñar una aplicación que se adaptara a las necesidades del usuario, en este caso, el propietario del camping. La interfaz ofrecida debía ser intuitiva, con acciones claras y retroalimentación instantánea. También era importante hacer un uso eficiente de los recursos, ya que el usuario no puede esperar demasiado a que se ejecuten sus acciones. El diseño debía ser sencillo, no sobrecargado, accesible y consistente. Debe garantizar una navegación clara, con interfaces visualmente agradables.

4. REQUISITOS

4.1. CATÁLOGO DE REQUISITOS

REQUISITOS FUNCIONALES

Código	Descripción
RF-1	El sistema debe permitir al propietario crear una parcela.
RF-2	El sistema debe permitir al propietario modificar una parcela previamente creada.
RF-3	El sistema debe permitir al propietario eliminar una parcela previamente creada.
RF-4	El sistema debe permitir al propietario consultar el listado de las parcelas creadas.
RF-4.1	El sistema debe permitir al propietario ordenar el listado de las parcelas por identificador, número máximo de ocupantes o precio.
RF-5	El sistema debe permitir al propietario hacer una reserva.
RF-6	El sistema debe permitir al propietario modificar una reserva.
RF-7	El sistema debe permitir al propietario eliminar una reserva.
RF-8	El sistema debe permitir al propietario consultar el listado de reservas.
RF-8.1	El sistema debe permitir al propietario ordenar el listado de reservas por nombre, número móvil o fecha de entrada.
RF-9	El sistema debe comprobar que la reserva es válida.
RF-9.1	El sistema debe comprobar que la fecha de entrada de la reserva sea igual o posterior a la del día actual y la fecha de salida es posterior a la de entrada
RF-9.2	El sistema debe comprobar que no se producen solapes en las fechas de reserva de las parcelas.
RF-9.3	El sistema debe comprobar que no se supera la capacidad máxima de ocupantes de la parcela.
RF-10	El sistema debe calcular automáticamente el precio total de una reserva.
RF-11	El sistema debe enviar al móvil del cliente la información de la reserva cuando el propietario lo vea oportuno.

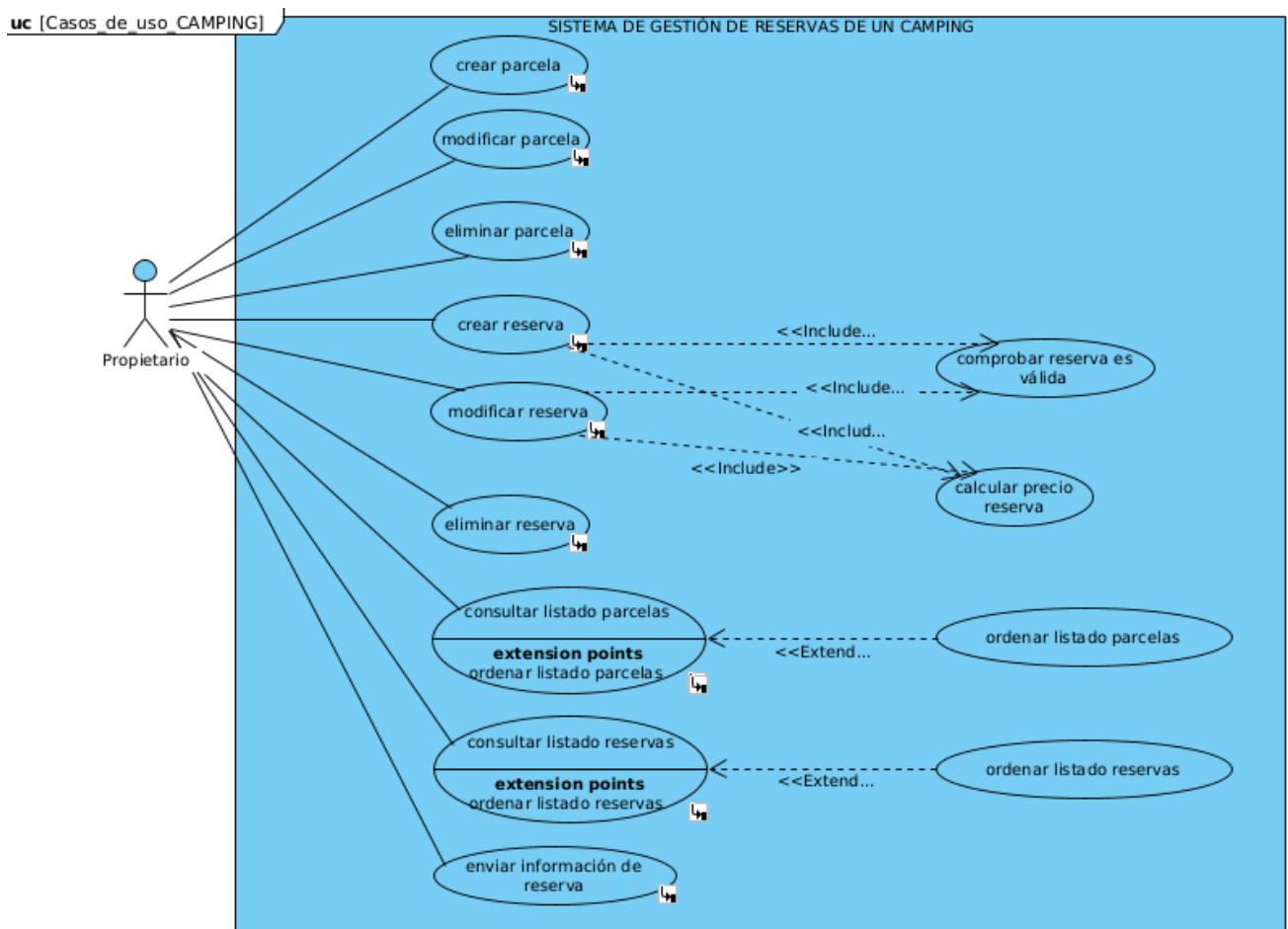
REQUISITOS NO FUNCIONALES

Código	Descripción
RNF-1	El sistema permite como máximo 100 parcelas.
RNF-2	El sistema permite como máximo 10.000 reservas.
RNF-3	El sistema debe estar desarrollado para Android.

DICCIONARIO DE DATOS

- **Parcela:** una parcela se identifica con el nombre de un monte o vall.e e incluye una descripción de sus características (tamaño, disponibilidad de agua, luz, número máximo de ocupantes y precio en euros por persona).
- **Reserva:** una reserva consta de un nombre de cliente, su número móvil, la fecha de entrada y salida, y las parcelas reservadas (incluyendo el número de ocupantes de cada una).

4.2. DIAGRAMA DE CASOS DE USO



Caso de uso “crear parcela”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción crear parcela.
- 2) El sistema solicita al propietario que introduzca el nombre que identifica a la parcela, el tamaño, la disponibilidad de agua y luz, el número máximo de ocupantes y el precio por persona en euros.
- 3) El propietario introduce los datos solicitados de la parcela.
- 4) El sistema confirma al usuario que la parcela se ha creado correctamente.

Flujo de eventos alternativo:

- 1) El propietario no introduce todos los datos de la parcela solicitados por el sistema.
- 2) El sistema le informa al propietario de que hay un error en los datos introducidos y finaliza el caso de uso.

Flujo de eventos alternativo:

- 1) El propietario introduce los datos de una parcela que ya existe.
- 2) El sistema le informa al usuario de que hay un error al crear una parcela ya existente y finaliza el caso de uso.

Caso de uso “modificar parcela”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de modificar una parcela.
- 2) El sistema le solicita al propietario el identificador de la parcela que desea modificar.
- 3) El propietario introduce el identificador de la parcela a modificar.
- 4) El sistema le solicita al propietario los nuevos datos de la parcela (tamaño, disponibilidad de agua y luz, número máximo de ocupantes y precio por persona en euros).
- 5) El propietario introduce la nueva información de la parcela.
- 6) El sistema confirma al propietario que la parcela ha sido modificada correctamente.

Flujo de eventos alternativo:

- 1) El usuario introduce el identificador de una parcela que no existe en el sistema.
- 2) El sistema le informa al propietario que la parcela que desea modificar no está registrada en el sistema y finaliza el caso de uso.

Flujo de eventos alternativo:

- 1) El propietario no introduce todos los nuevos datos de la parcela que quiere modificar.
- 2) El sistema le informa al propietario de que debe introducir todos los nuevos datos y finaliza el caso de uso.

Caso de uso “eliminar parcela”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de eliminar la parcela.
- 2) El sistema solicita al usuario que introduzca el identificador de la parcela que desea eliminar.
- 3) El propietario introduce el identificador de la parcela a eliminar.
- 4) El sistema le informa al usuario de que la parcela ha sido eliminada correctamente.

Flujo de eventos alternativo:

- 1) El propietario no introduce el identificador de la parcela que desea eliminar.
- 2) El sistema le informa al propietario de que debe introducir el identificador de la parcela y finaliza el caso de uso.

Flujo de eventos alternativo:

- 3) El propietario introduce el identificador de una parcela que no existe.
- 4) El sistema le informa al propietario de que la parcela que desea eliminar no existe en el sistema y finaliza el caso de uso.

Caso de uso “crear reserva”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de crear una reserva.
- 2) El sistema solicita al propietario el nombre del cliente y su número móvil, además de la fecha de entrada y salida, y las parcelas a reservar, incluyendo el número de ocupantes de cada una.
- 3) El propietario introduce los datos solicitados.
- 4) **include** “comprobar reserva válida”.
- 5) **include** “calcular precio reserva”.
- 6) El sistema informa al propietario de que la reforma ha sido creada correctamente.

Flujo de eventos alternativo:

- 1) El usuario introduce no introduce todos los datos que le ha solicitado el sistema.
- 2) El sistema informa al usuario del error y finaliza el caso de uso.

Flujo de eventos alternativo:

- 1) El usuario introduce los datos de una reserva que ya existe en el sistema (las parcelas a reservar ya están reservadas en las fechas introducidas por el propietario).
- 2) El sistema informa al usuario del error y finaliza el caso de uso.

Flujo de eventos alternativo:

- 1) El usuario introduce los datos de una reserva y esta no es válida.
- 2) El sistema informa al usuario del error y finaliza el caso de uso.

Caso de uso “modificar reserva”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de modificar una reserva.
- 2) El sistema le solicita al propietario los datos de la reserva que desea modificar: la fecha de entrada y salida, y las parcelas reservadas.
- 3) El propietario introduce los datos solicitados.
- 4) El sistema le solicita al propietario los nuevos datos de la reserva: nombre del cliente, número móvil, fecha de entrada, fecha de salida, parcelas y sus ocupantes.
- 5) El propietario introduce los datos de la nueva reserva.
- 6) **include** “comprobar reserva válida”.
- 7) **include** “calcular precio reserva”.
- 8) El sistema le informa al propietario de la correcta modificación de la reserva.

Flujo de eventos alternativo:

- 1) El propietario introduce los datos de una reserva que no existe en el sistema.
- 2) El sistema le informa del error y finaliza el caso de uso.

Flujo de eventos alternativo:

- 1) El propietario no introduce todos los datos de la nueva reserva solicitados por el sistema.
- 2) El sistema le informa del error y finaliza el caso de uso.

Flujo de eventos alternativo:

- 1) El propietario introduce los nuevos datos de una reserva que no es válida.
- 2) El sistema le informa del error y finaliza el caso de uso.

Caso de uso “eliminar reserva”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de eliminar una reserva.
- 2) El sistema solicita al usuario los datos de la reserva que desea eliminar: la fecha de entrada y salida junto con las parcelas reservadas.
- 3) El propietario introduce los datos solicitados sobre la reserva que desea eliminar.
- 4) El sistema informa al usuario de que la reserva ha sido eliminada correctamente.

Flujo de eventos alternativo:

- 1) El propietario no introduce todos los datos solicitados por el sistema para eliminar la reserva.
- 2) El sistema informa al propietario del error y finaliza el caso de uso.

Flujo de eventos alternativo:

- 1) El propietario introduce los datos de la reserva que desea eliminar y estos no están registrados en el sistema.
- 2) El sistema informa al propietario del error y finaliza el caso de uso.

Caso de uso “comprobar reserva válida”:

Flujo de eventos principal:

- 1) El sistema comprueba que la fecha de entrada de la reserva sea igual o posterior a la del día actual.
- 2) El sistema comprueba que la fecha de salida sea posterior a la fecha de entrada.
- 3) El sistema comprueba que no se supere la capacidad máxima de ocupantes de cada parcela de la reserva.
- 4) El sistema comprueba que no haya solapes en la reserva de parcelas, es decir, que las parcelas reservadas no estén previamente reservadas por cualquier otra reserva en las fechas introducidas.

Caso de uso “calcular precio reserva”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario ha creado una reserva.
- 2) El sistema calcula el precio de la reserva, sumando el producto del número de ocupantes de cada parcela y el precio por persona de dicha parcela.
- 3) El sistema informa al propietario del precio de la reserva.

Flujo de eventos alternativo:

- 1) El caso de uso comienza cuando el propietario ha modificado la selección de parcelas o el número de ocupantes de la parcela.
- 4) El sistema recalcula el precio de la reserva, sin cambiar el precio por persona de las parcelas.
- 5) El sistema informa al propietario del nuevo precio de la reserva.

Caso de uso “consultar listado parcelas”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de consultar el listado de parcelas.
- 2) El sistema pregunta al propietario si desea ordenar el listado de parcelas.
- 3) El propietario selecciona si desea ordenar el listado.
- 4) **extend** “ordenar listado parcelas”.
- 5) El sistema muestra al propietario todas las parcelas existentes junto con sus correspondientes datos (nombre, tamaño, disponibilidad de agua, luz, número máximo de ocupantes y precio en euros por persona).

Caso de uso “ordenar listado parcelas”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario elige ordenar el listado de parcelas.
- 2) El sistema pregunta al propietario si desea ordenar el listado según el identificador, el número máximo de ocupantes o el precio.
- 3) El propietario selecciona el criterio de ordenación.

Caso de uso “consultar listado reservas”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de consultar el listado de reservas.
- 2) El sistema pregunta al propietario si desea ordenar el listado de reservas
- 3) El propietario selecciona si desea ordenar el listado.
- 4) **extend** “ordenar listado reservas”.
- 5) El sistema muestra al usuario todas las reservas existentes junto con sus correspondientes datos (nombre del cliente, número móvil y fecha de entrada).

Caso de uso “ordenar listado reservas”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario elige ordenar el listado de reservas.
- 2) El sistema pregunta al propietario si desea ordenar el listado según el nombre, el número de móvil o la fecha de entrada.
- 3) El propietario selecciona el criterio de ordenación.

Caso de uso “enviar información de reserva”:

Flujo de eventos principal:

- 1) El caso de uso comienza cuando el propietario selecciona la opción de enviar la información de la reserva al móvil del cliente.
- 2) El sistema solicita al propietario los datos de la reserva que desea enviar al cliente.
- 3) El propietario introduce los datos de la reserva.
- 4) El sistema envía al móvil del cliente registrado en la reserva todos los datos.
- 5) El sistema avisa al propietario de que los datos de la reserva han sido enviados correctamente.

Flujo de eventos alternativo:

- 1) El propietario no introduce todos los datos solicitados por el sistema para localizar la reserva.
- 2) El sistema informa al propietario del error y finaliza el caso de uso.

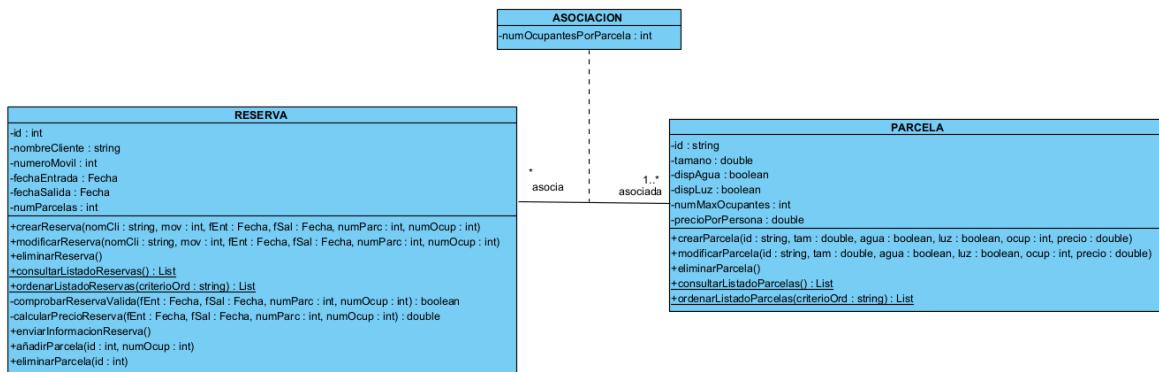
Flujo de eventos alternativo:

- 1) El propietario introduce los datos de la reserva que desea enviar al cliente pero esta no existe.
- 2) El sistema informa al propietario del error y finaliza el caso de uso.

5. ANÁLISIS

5.1 DIAGRAMA DE CLASES

A continuación, presentamos el diagrama de clases generado, en el cual se representan las clases "Parcela" y "Reserva". Además, se puede observar la clase de asociación que resulta de la relación entre estas dos clases.



5.2 DIAGRAMAS DE SECUENCIA

Seguidamente, mostramos una serie de diagramas UML que detallan cada uno de los casos de uso identificados en nuestro diagrama de casos de uso original.

Estas imágenes nos permiten apreciar con mayor claridad cómo interactúan el propietario, el sistema y las clases implicadas en cada caso, proporcionando una visión más profunda de la dinámica y las relaciones entre los elementos del sistema.

Diagrama de secuencia del caso de uso “Crear parcela”

En este diagrama, se crea la parcela con la información indicada de nombre, tamaño, disponibilidad de agua y luz, y precio.

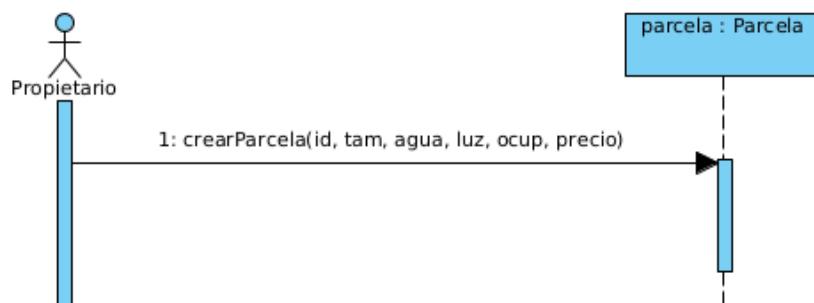


Diagrama de secuencia del caso de uso “Modificar parcela”

En este diagrama, se modifica la parcela previamente creada, cambiando sus datos por los introducidos por el propietario.

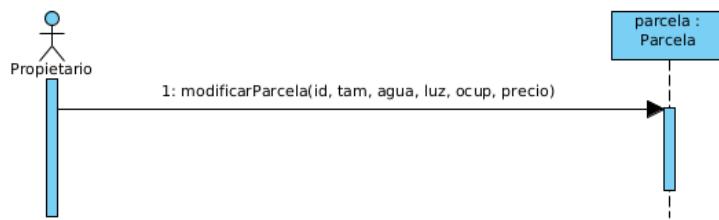


Diagrama de secuencia del caso de uso “Eliminar parcela”

En este diagrama, se elimina la parcela cuyo identificador es igual al introducido por el propietario.

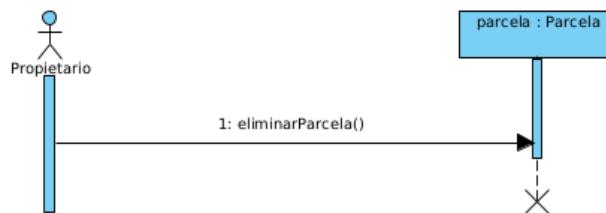


Diagrama de secuencia del caso de uso “Crear reserva”

En este diagrama, creamos una reserva con los datos introducidos por el usuario (nombre del cliente, número móvil, fecha de entrada, fecha de salida, número de parcelas y número de ocupantes por parcela). Se crea una parcela por cada una que el propietario desea reservar. Por último, se comprueba que la reserva sea válida y se calcula el precio total de la reserva.

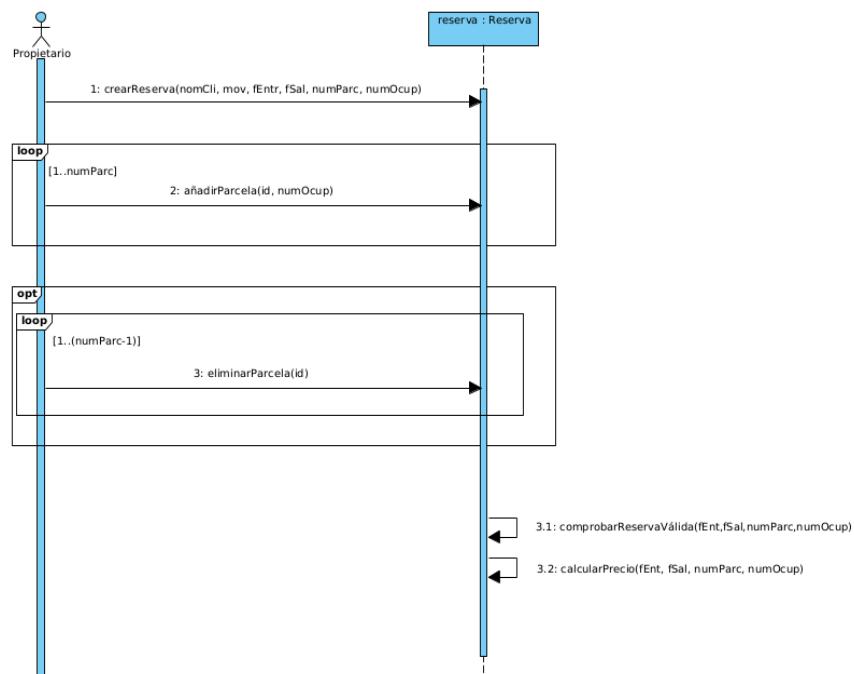


Diagrama de secuencia del caso de uso “Modificar reserva”

En este diagrama, si el usuario desea añadir más parcelas de las previamente reservadas, se añadirán. Sin embargo, si desea reservar menos parcelas, se eliminarán las que desea eliminar. Por último, se comprobará que la reserva modificada sea válida y se calculará el precio de la reserva.

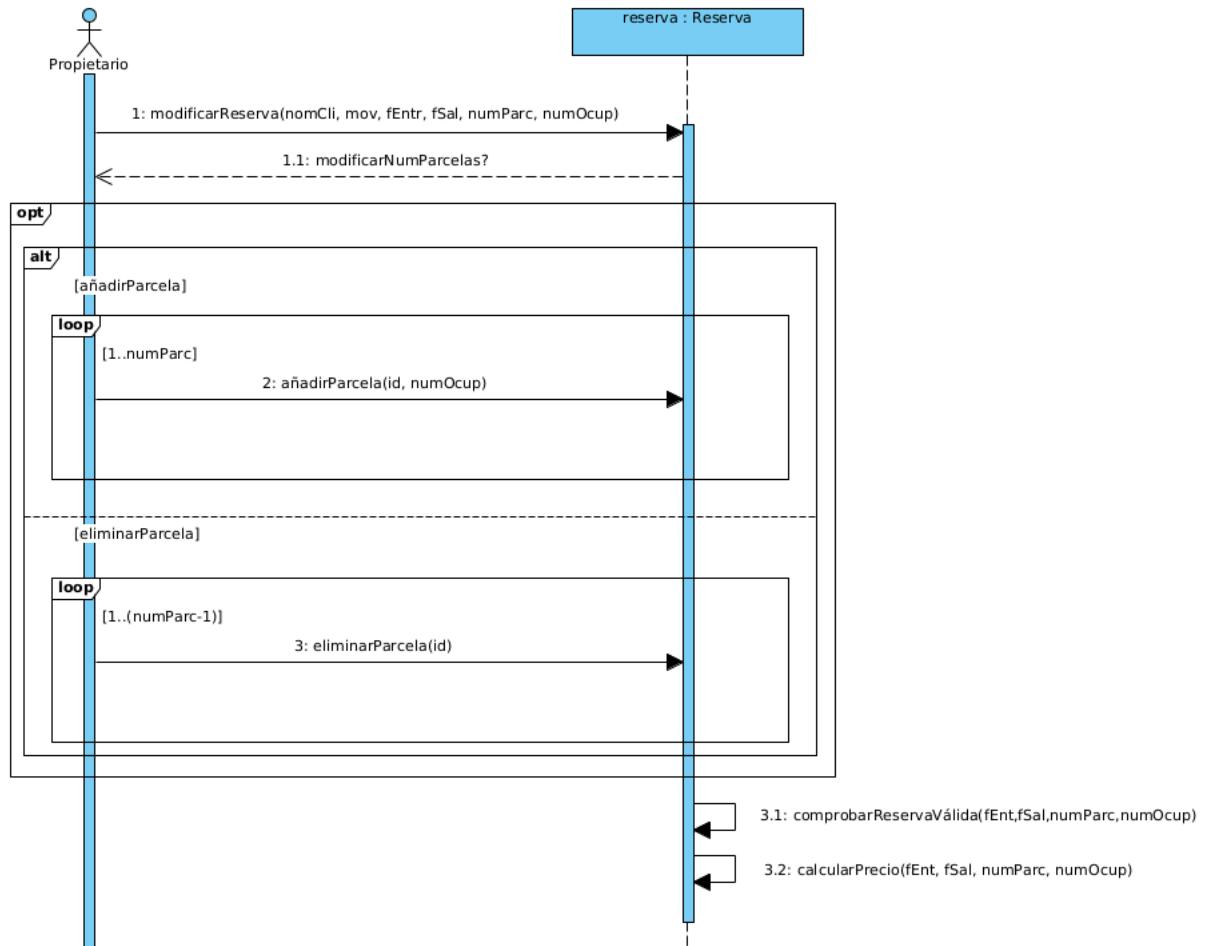


Diagrama de secuencia del caso de uso “Eliminar reserva”

En este diagrama, se eliminará la reserva que tenga el identificador introducido por el propietario.

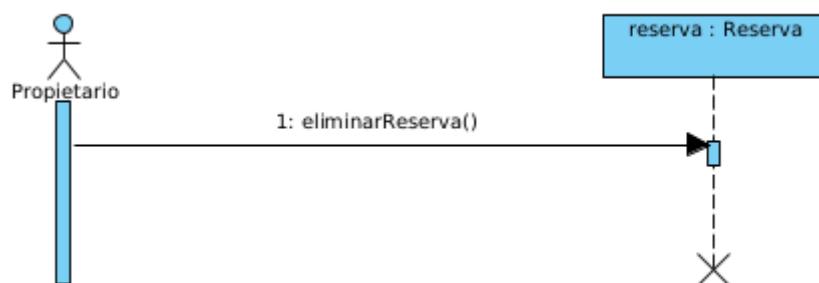


Diagrama de secuencia del caso de uso “Consultar listado de parcelas”

En este diagrama, el propietario desea consultar el listado de las parcelas, y se le pregunta si desea ordenarlo. En base a su respuesta, se llevará a cabo el caso de uso de ordenar listado de parcelas (si la respuesta es afirmativa), y si no, mostrará la información de cada parcela que esté registrada.

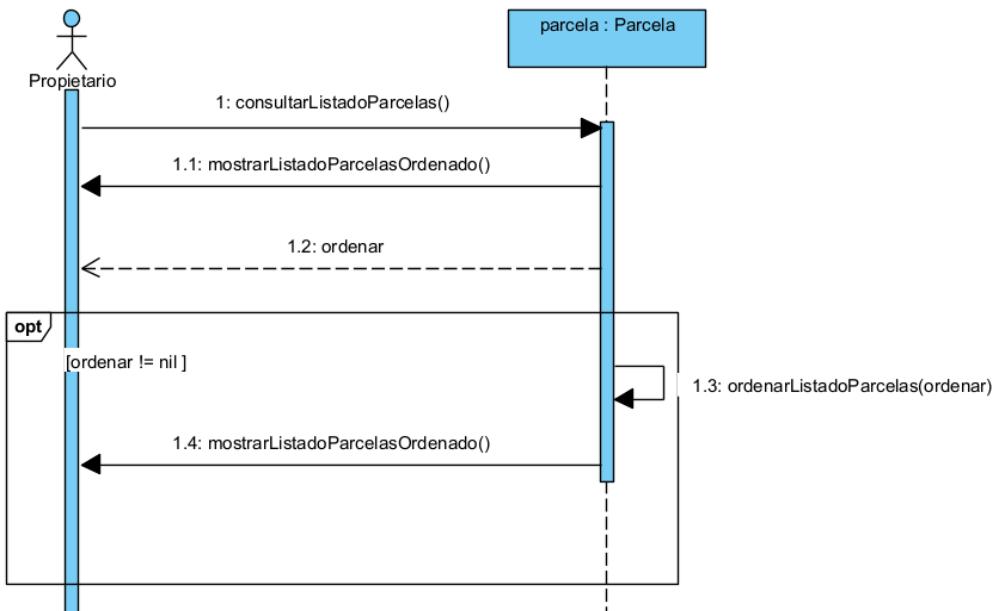


Diagrama de secuencia del caso de uso “Consultar listado reservas”

En este diagrama, el propietario desea consultar el listado de las reservas, y se le pregunta si desea ordenarlo. En base a su respuesta, se llevará a cabo el caso de uso de ordenar listado de reservas (si la respuesta es afirmativa), y si no, mostrará la información de cada reserva que esté registrada.

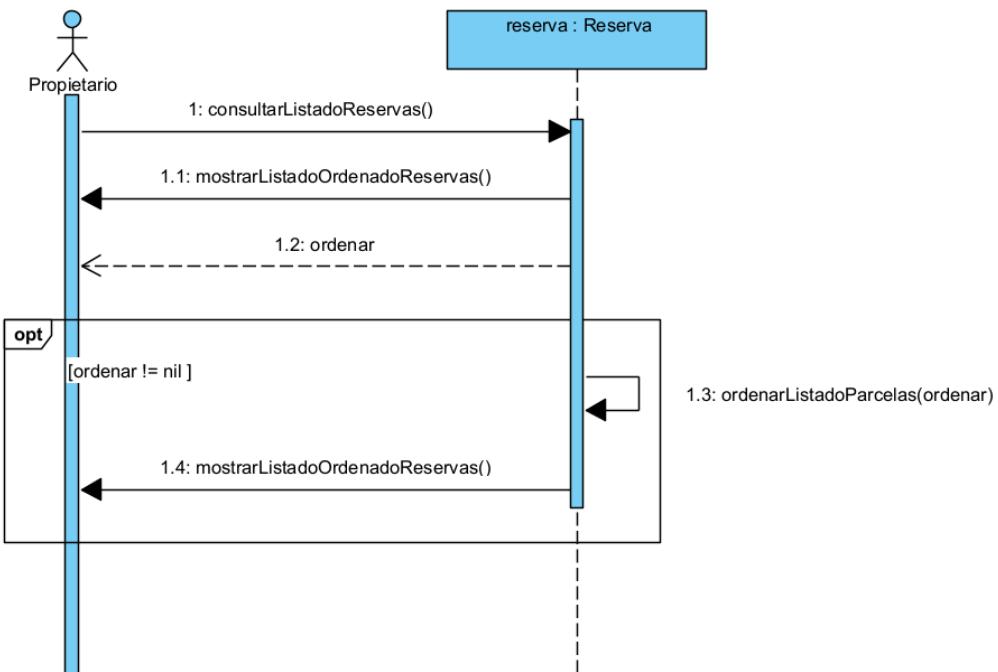
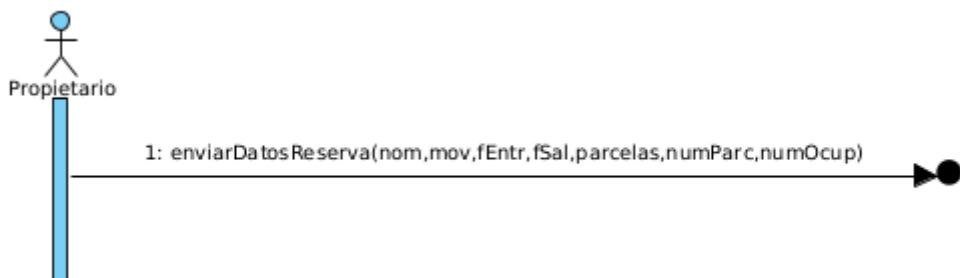


Diagrama de secuencia del caso de uso “Enviar información reserva”

En este diagrama, la situación comienza cuando el propietario desea enviar la información de la reserva al cliente. Se le solicitan los datos de la reserva en cuestión, y se comprueba que no sean nulos. Luego, si existe esa reserva, se le envían los datos finalmente al cliente y se le notifica al propietario con un mensaje de feedback, si no, envía un mensaje de error de que la reserva no existe.



6. PROTOTIPOS DE PANTALLAS

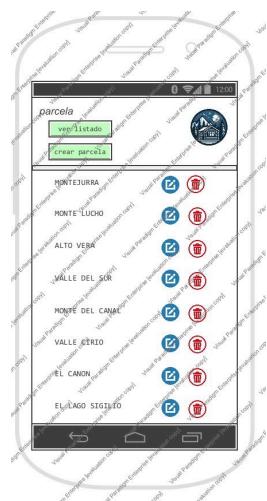
Pantalla de selección inicial:

- Al abrir la aplicación, se presenta una pantalla con dos opciones:
 - **Parcelas**
 - **Reservas**
- El usuario debe seleccionar una de estas opciones para continuar.



Si se selecciona "Parcelas":

- Se abre una nueva pantalla con las siguientes acciones disponibles:
 - **Crear Parcela** (texto botón)
 - **Modificar Parcela** (ícono botón)
 - **Eliminar Parcela** (ícono botón)
 - **Mostrar Listado de Parcelas** (texto botón)



Crear Parcela:

- Al seleccionar "Crear Parcela", se muestra una pantalla donde se deben completar varios campos obligatorios para crear una nueva parcela.
- Si algún campo no se rellena, aparece un mensaje de error en rojo y se resaltan en rojo los campos faltantes.
- Una vez todos los campos están correctamente completados, la parcela se crea.



Modificar Parcela:

- Si se selecciona "Modificar Parcela", se muestra la misma pantalla utilizada en "Crear Parcela", pero con los datos ya llenados de la parcela a modificar.
- El usuario puede modificar cualquier campo y confirmar los cambios.



Eliminar Parcela:

- Se presenta un listado con todas las parcelas disponibles, con la posibilidad de hacer scroll vertical para ver todas las opciones.
- Al lado de cada parcela, aparece un icono de papelera que, al pulsarlo, elimina la parcela seleccionada.



Mostrar Listado de Parcelas:

- Aparece una tabla con todas las parcelas.
- En la parte superior de la tabla, se pueden seleccionar opciones para ordenar por: **identificador, número máximo de habitantes y precio**

PARCELA	ID	PRECIO	MAX_OCUP
MONTEJURRA	1	20€	6
MONTE LUCHO	1	60€	8
ALTO VERA	1	15€	2
VALLE DEL BUR	1	20€	5
MONTE DEL CANÁL	1	50€	5
VALLE CIRIO	1	80€	10
EL CANÓN	1	25€	4

Si se selecciona "Reservas":

- Se abre una pantalla con las siguientes opciones:
 - **Crear Reserva (texto botón)**
 - **Modificar Reserva (ícono botón)**
 - **Eliminar Reserva (ícono botón)**
 - **Mostrar Listado de Reservas (texto botón)**
 - **Enviar información al cliente sobre la reserva (ícono botón)**



Crear Reserva:

- Al seleccionar "Crear Reserva", se muestra una pantalla donde se deben completar varios campos obligatorios.
- Es posible agregar varias parcelas a la misma reserva pulsando el símbolo "+" que permite añadir nuevos campos para otras parcelas.
- Si falta algún campo por llenar o hay un error (fecha incorrecta, parcela no disponible, número de ocupantes excedido), se muestra un mensaje de error.



Modificar Reserva:

- Al seleccionar "Modificar Reserva", se presenta una pantalla similar a "Crear Reserva", pero con los datos de la reserva ya completados.
- El usuario puede modificar los campos necesarios y confirmar los cambios.

The diagram illustrates the 'Modificar Reserva' (Modify Reservation) process across five mobile device screens. The top row shows the initial state and a validation step. The bottom row shows three error cases where specific input constraints are violated.

Initial State (Top Left):

- Fields: Nombre: Juan Pérez, DNI: 63689276, Fecha entrada: 8/7/2024, Fecha salida: 11/7/2024, Parcela 1: MONTEJURRA 5, Parcela 2: ALTO VERA 8.
- Message: *complete todos los campos
- Action: Confirmar cambios

Validation Step (Top Right):

- Fields: Same as initial state.
- Message: *complete todos los campos
- Action: Confirmar cambios

Error Case 1 (Bottom Left):

- Fields: Same as initial state.
- Message: SE HA SUPERADO LA CAPACIDAD MÁXIMA DE LA PARCELA
- Action: Confirmar cambios

Error Case 2 (Bottom Middle):

- Fields: Same as initial state.
- Message: LA PARCELA NO ESTÁ DISPONIBLE PARA ESAS FECHAS
- Action: Confirmar cambios

Error Case 3 (Bottom Right):

- Fields: Same as initial state.
- Message: LAS FECHAS INTRODUCIDAS SON INVÁLIDAS
- Action: Confirmar cambios

Eliminar Reserva:

- Se muestra una lista de todas las reservas disponibles, con la opción de hacer scroll vertical.
- Al lado de cada reserva hay un ícono de papelera, que al pulsarlo, elimina la reserva seleccionada.



Mostrar Listado de Reservas:

- Se presenta una tabla con todas las reservas.
- En la parte superior de la tabla, se puede seleccionar si se quiere ordenar por: **s**

The image shows a mobile application interface displaying a list of reservations. At the top, it says 'reserva' and 'LISTADO'. Below this is a text input field labeled 'Ordenar por:' with the letter 's' typed into it. The main area contains a table with the following data:

Nº	NOMBRE	TELÉFONO	ENTRADA
1	Juan Pérez	636895142	07-11-25
2	Maria Pio	656985245	12-07-25
3	Lucía Sis	6367512483	10-08-26
4	Carla Montañés	698123658	06-06-25
5	Carlos Gáñarul	614596866	24-12-24
6	Nicolás Agreda	636599989	25-07-25
7	Carlota Agreda	654898977	04-11-28

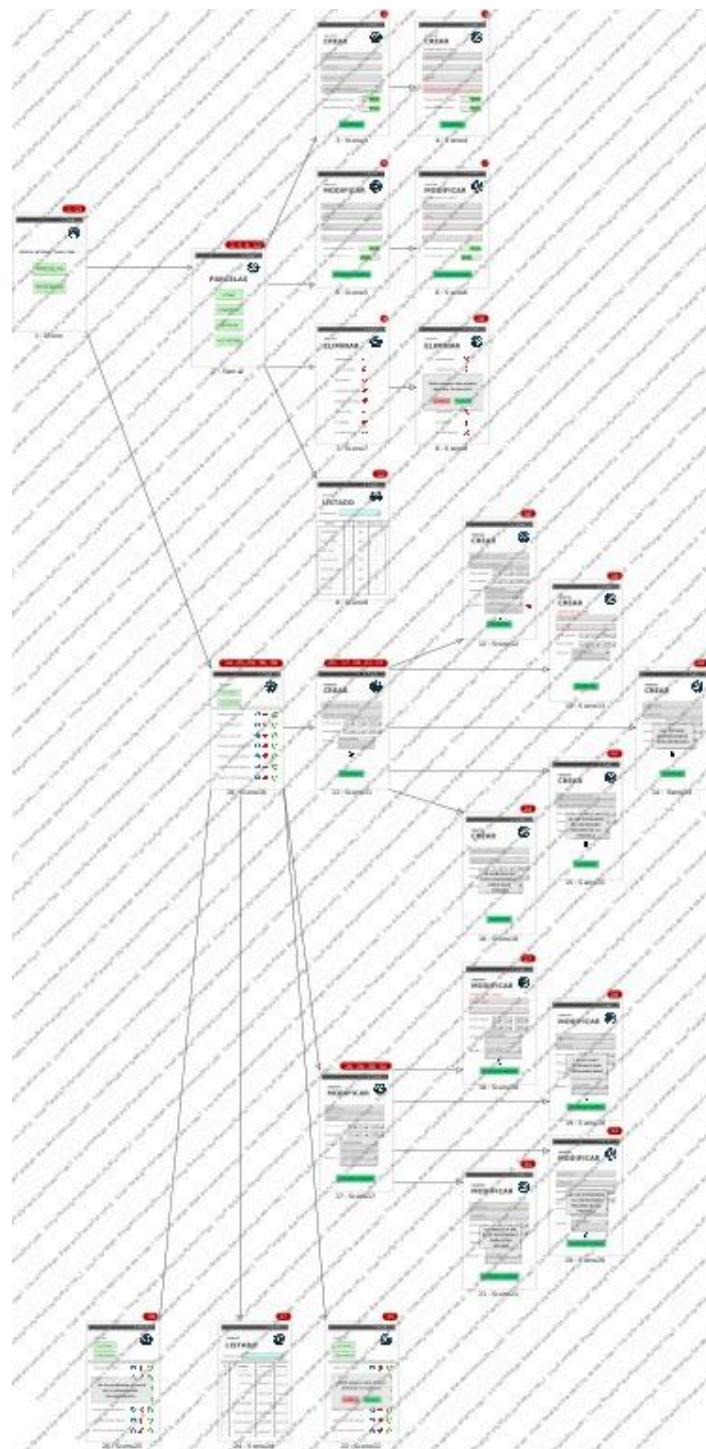
Enviar información al cliente:

- Aparece un listado con todas las reservas.
- Al lado de cada reserva hay un ícono de "enviar".
- Al presionar este ícono, la información de la reserva se envía automáticamente al cliente y se notifica al usuario que el envío se ha realizado correctamente.



7. MAPA DE NAVEGACIÓN

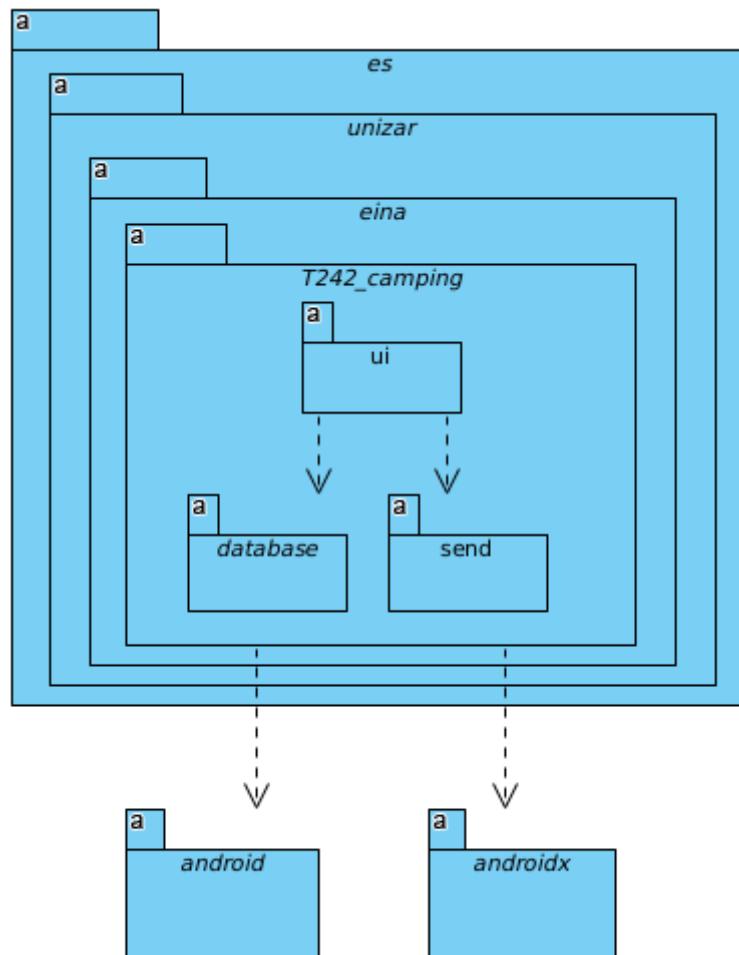
Hemos realizado un mapa de navegación desde visual paradigm, y hemos puesto un orden que se puede visualizar a modo vídeo de cómo sería la interacción con la aplicación y qué pantallas visualizará el propietario (suponiendo que comete todos los errores al crear, modificar...). Además hemos efectuado otro mapa de navegación desde un dispositivo porque creemos que así es más fácil de comprender en profundidad.



8. DIAGRAMA DE PAQUETES

Primeramente, hemos definido tal y como se especifica en el enunciado el paquete **es.unizar.eina.T242_camping**, donde tenemos 2 subsistemas, database y ui. En database, pondremos posteriormente las clases que facilitan el acceso a la base de datos, mientras que en ui pondremos todo lo relacionado con la interfaz de usuario.

El diagrama de paquetes obtenido es el siguiente:

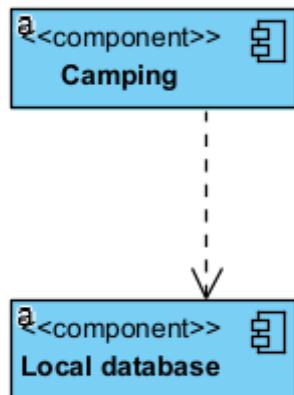


Tras esta definición, la estructura de carpetas de nuestro modelo queda tal que así:



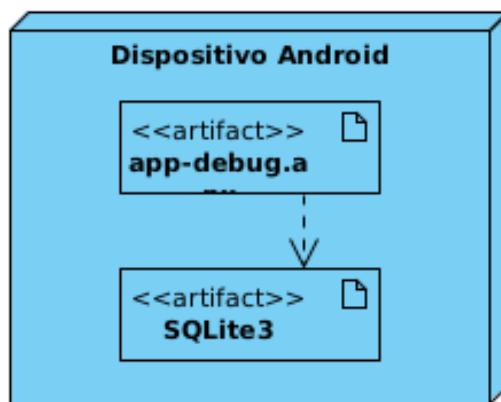
9. DIAGRAMA DE COMPONENTES

El diagrama de componentes es muy sencillo, ya que contamos únicamente con 2 componentes: la componente **Camping** que es la aplicación de gestión de reservas de las parcelas de un camping, y la componente **Local database** que representa la base de datos donde almacenamos las reservas y las parcelas.



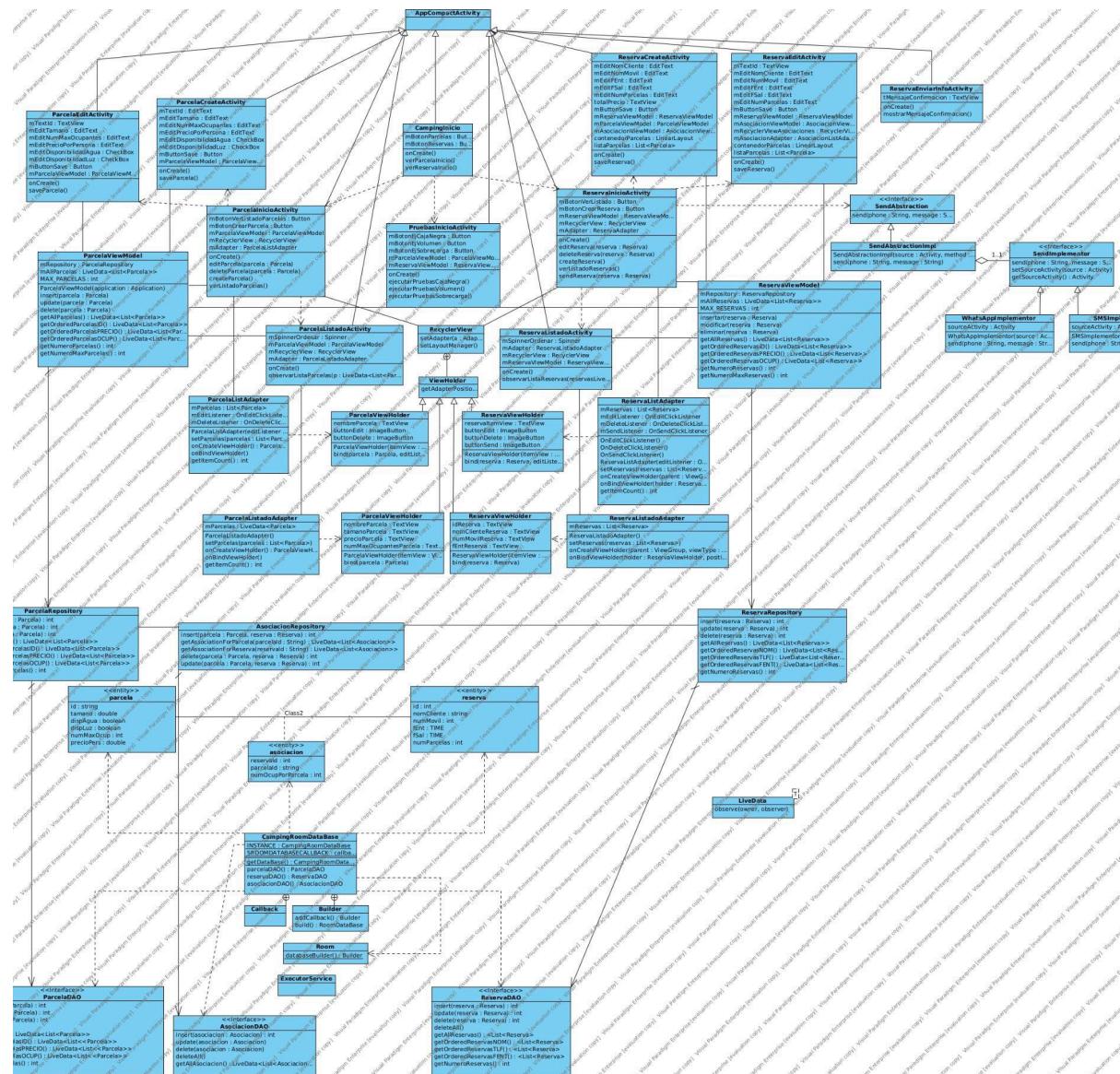
10. DIAGRAMA DE DESPLIEGUE

Para el diagrama de despliegue, comenzamos insertando el nodo o recurso computacional que en nuestro caso es cualquier **Dispositivo Android**. Continuamos añadiendo los 2 artefactos: **app-debug.apk** que es el ejecutable de la aplicación tras su compilación, el cual abriremos para utilizar la app, y **PostgreSQL** que es la tecnología que hemos utilizado para la base de datos. Ambos artefactos representan la versión física de los componentes Camping y Local database de nuestro diagrama de componentes.

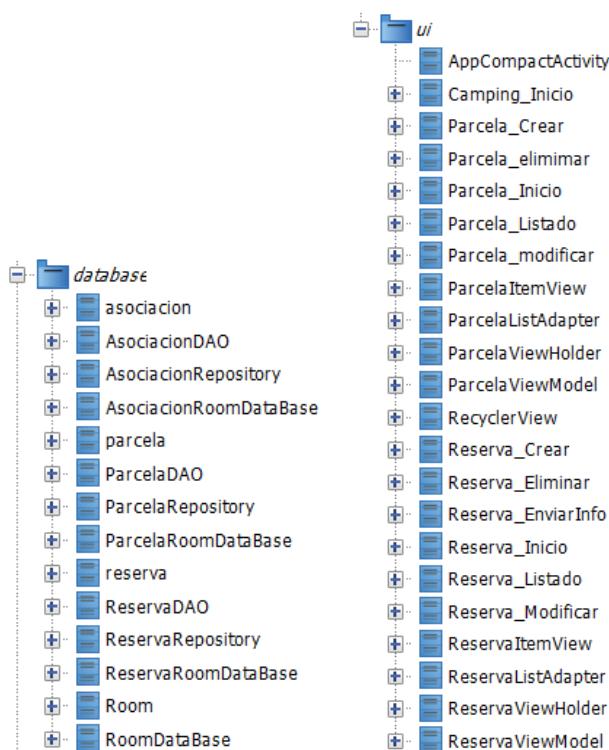


11. DIAGRAMA DE CLASES A NIVEL DE DISEÑO

Para crear el diagrama de clases, definimos primero las entidades tal cual nuestro modelo relacional: parcela, reserva y asociacion. A continuación, creamos la base de datos CampingRoomDatabase, y ya nos centramos en las clases que interactúan con la base de datos: DAO y Repository. Finalmente, definimos las clases relacionadas con la interfaz de usuario, siguiendo los prototipos de pantallas que habíamos definido previamente.



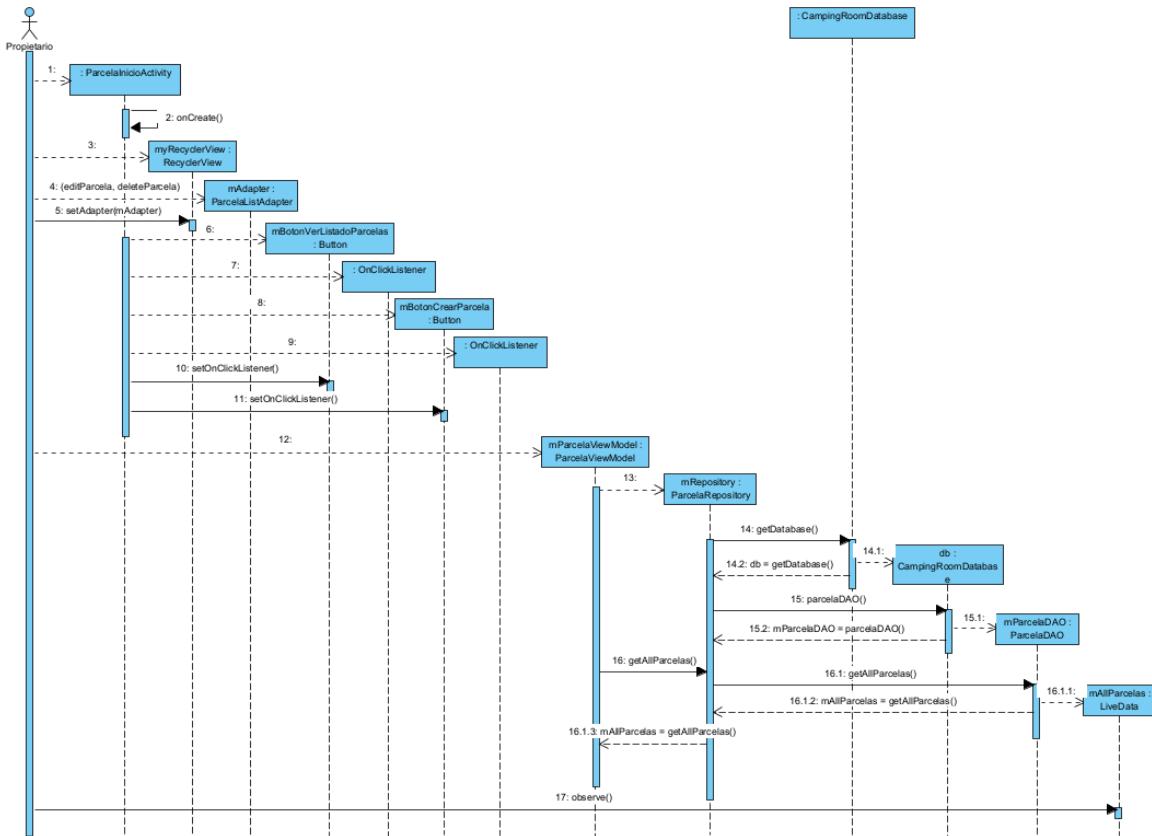
Tras crear el diagrama de clases, el esquema de directorios resultante es el siguiente:



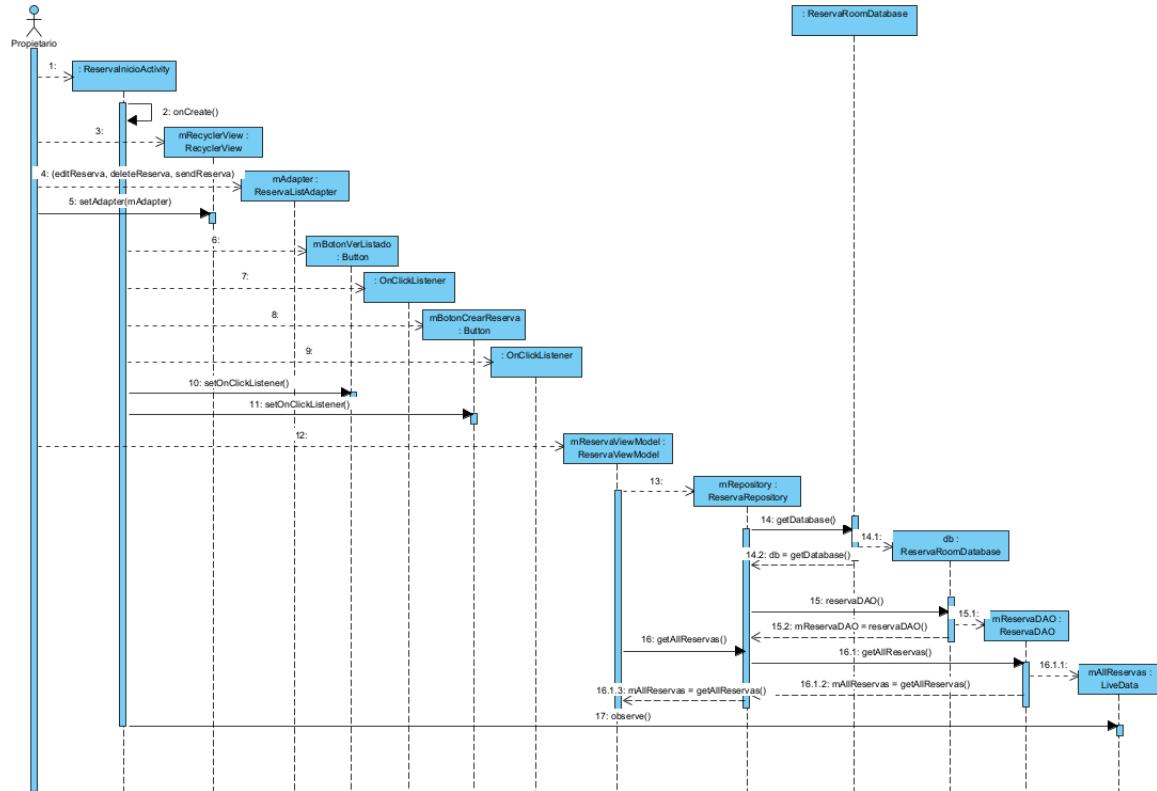
12. DIAGRAMAS DE SECUENCIA A NIVEL DE DISEÑO

Siguiendo el diagrama de clases a nivel de diseño, hemos realizado los diagramas de secuencia para cada una de las actividades (pantallas) de nuestra aplicación. En total, tenemos 10 diagramas de secuencia: **ParcelaInicioActivity**, **ParcelaListadoActivity**, **ParcelaCreateActivity**, **ParcelaEditActivity**, **ParcelaEliminar**, **ReservaInicioActivity**, **ReservaListadoActivity**, **ReservaCreateActivity**, **ReservaEditActivity** y **ReservaEliminar**.

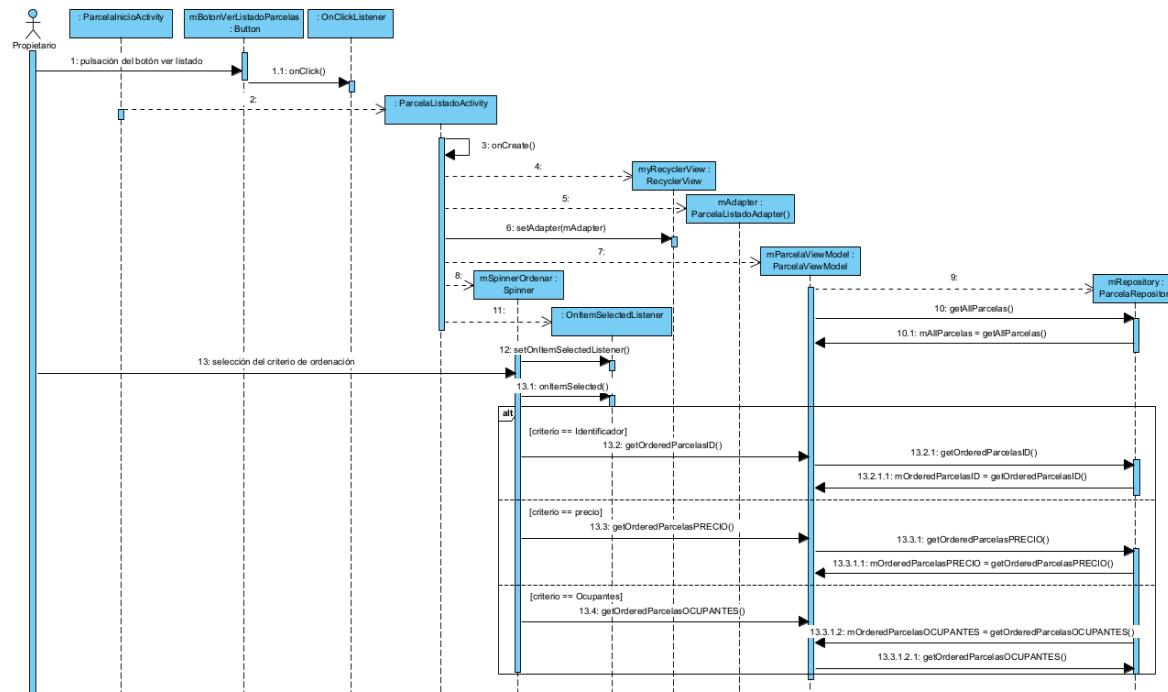
Para el de **ParcelaInicioActivity**, observamos cómo vamos creando secuencialmente las distintas clases necesarias para mostrar el listado de parcelas. Una vez llegamos a **ParcelaRepository**, éste es quien se encarga de solicitar las operaciones a la base de datos y obtener el listado de parcelas.



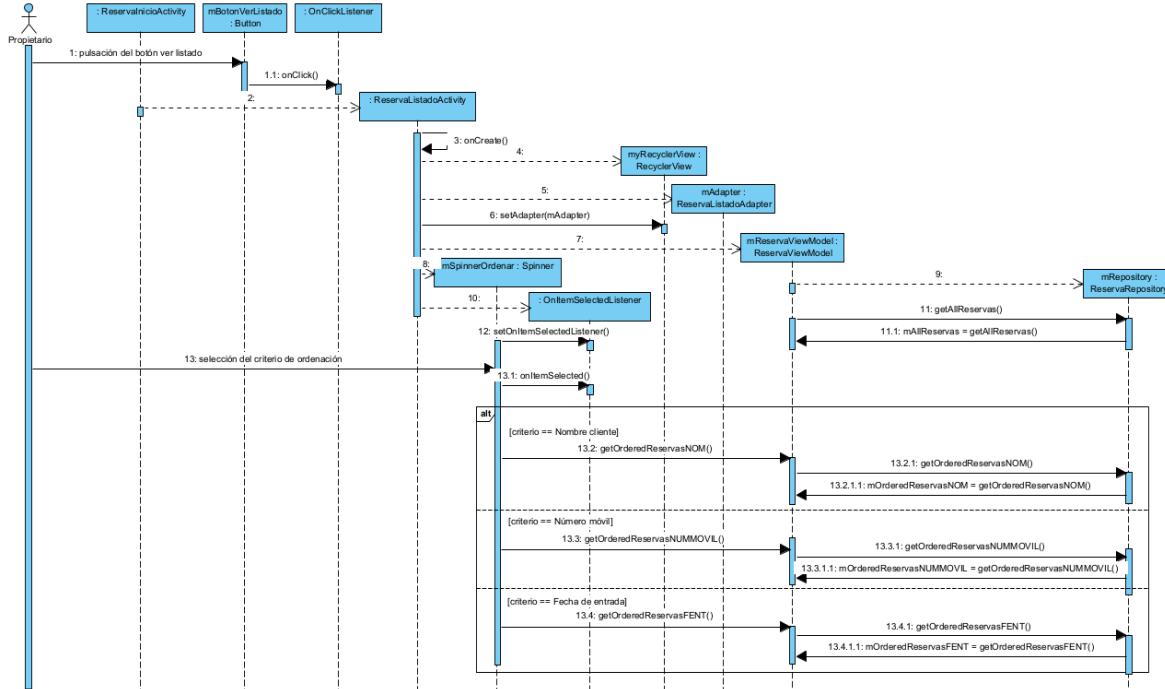
Por otro lado, tenemos el diagrama de secuencia de **ReservainicioActivity**, que es bastante similar al de ParcelainicioActivity, pero con las operaciones correspondientes de obtención del listado de reservas.



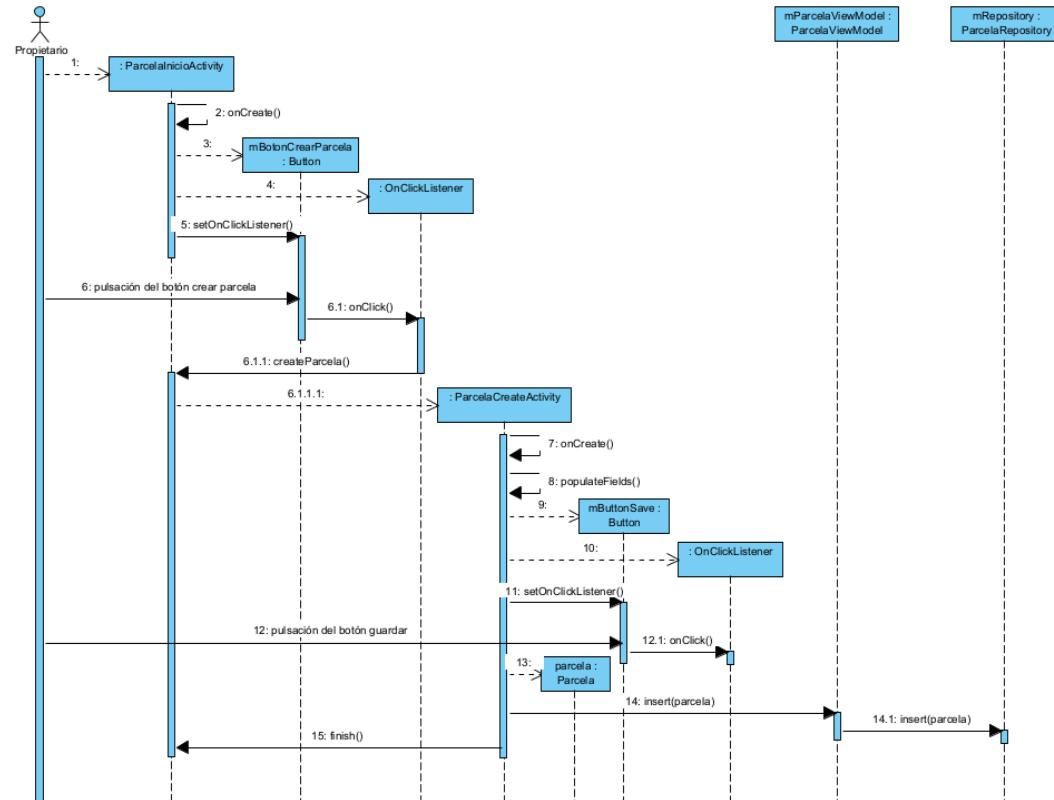
El diagrama de **ParcelaListadoActivity**, es similar a ParcelainicioActivity, ya que también muestra el listado de parcelas, lo único que muestra otros campos distintos, y no da la opción ni de editar ni de eliminar la parcela. En este caso, se da la opción de ordenar el listado de parcelas por Identificador, Precio o Número máximo de ocupantes.



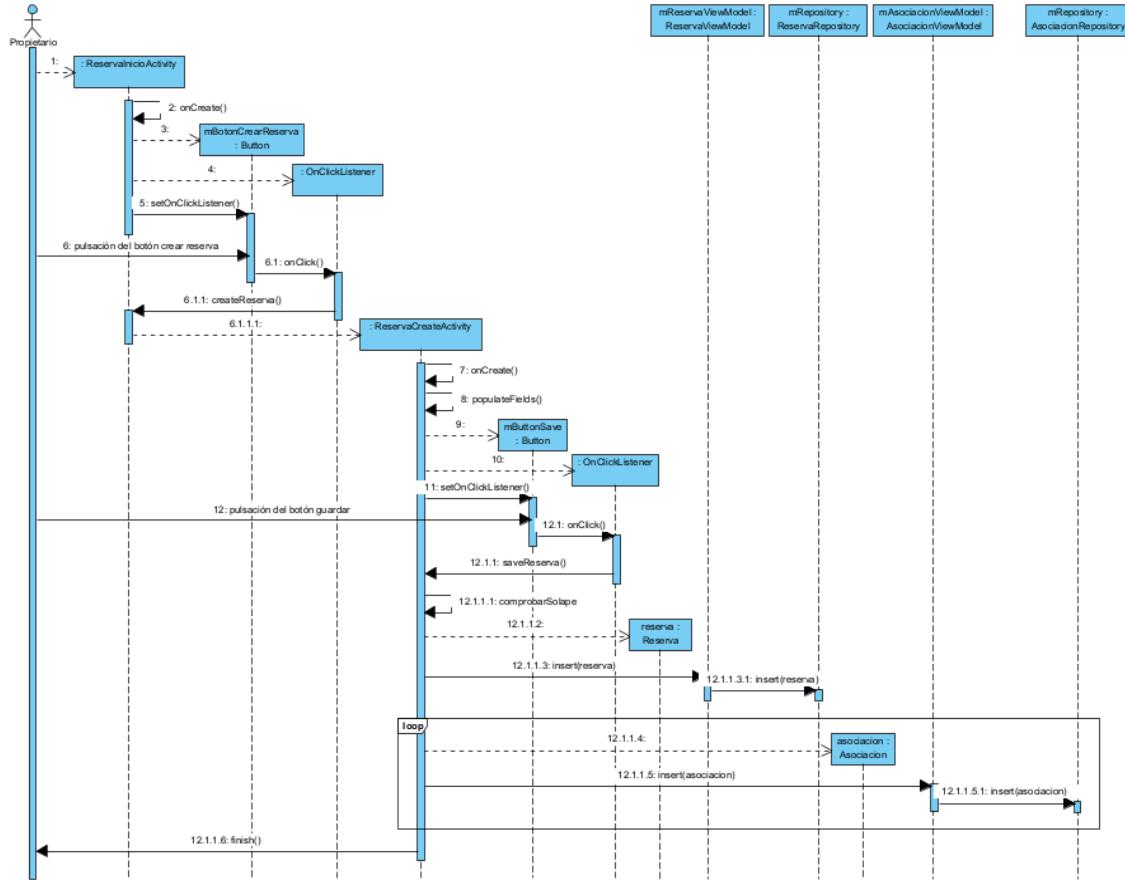
El diagrama de **ReservaListadoActivity** también es parecido al anterior ya que muestra el listado de parcelas del sistema con los campos: identificador, nombre del cliente, número móvil y fecha de entrada. También ofrece la posibilidad de ordenar el listado según el nombre del cliente, su número móvil o la fecha de entrada.



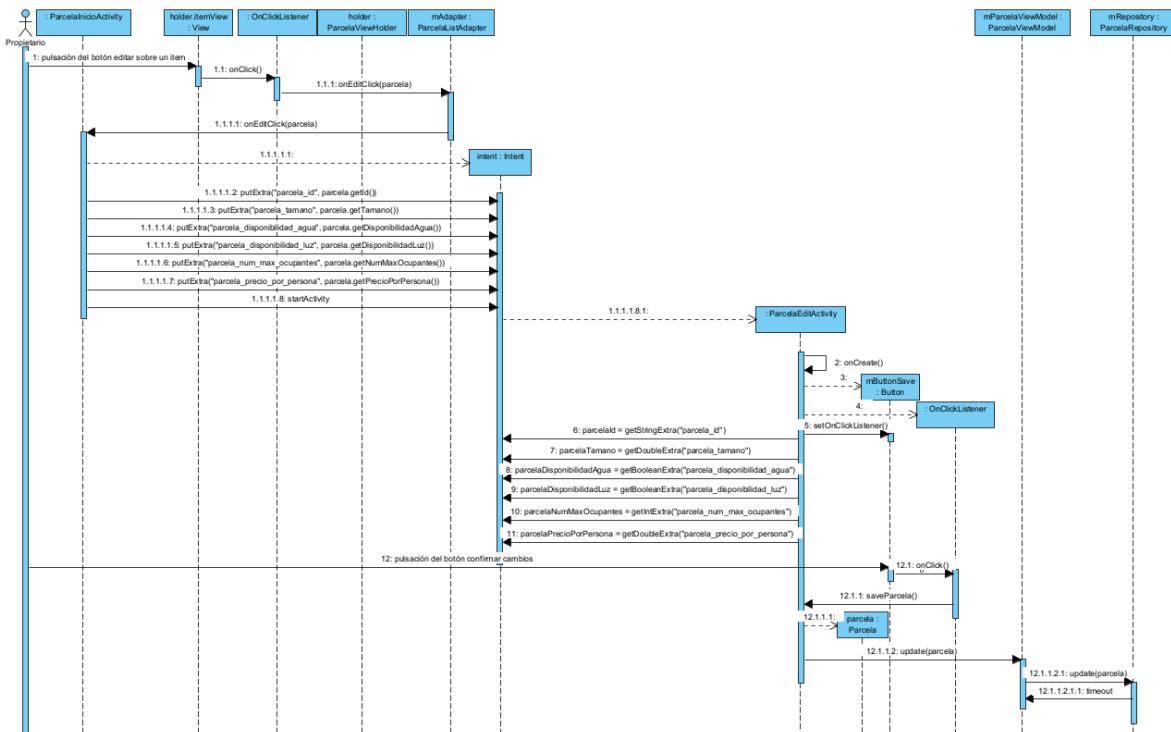
Para el caso de crear una parcela, el diagrama **ParcelaCreateActivity** muestra cómo se insertaría la parcela en la base de datos cuando el usuario pulse el botón de guardar.



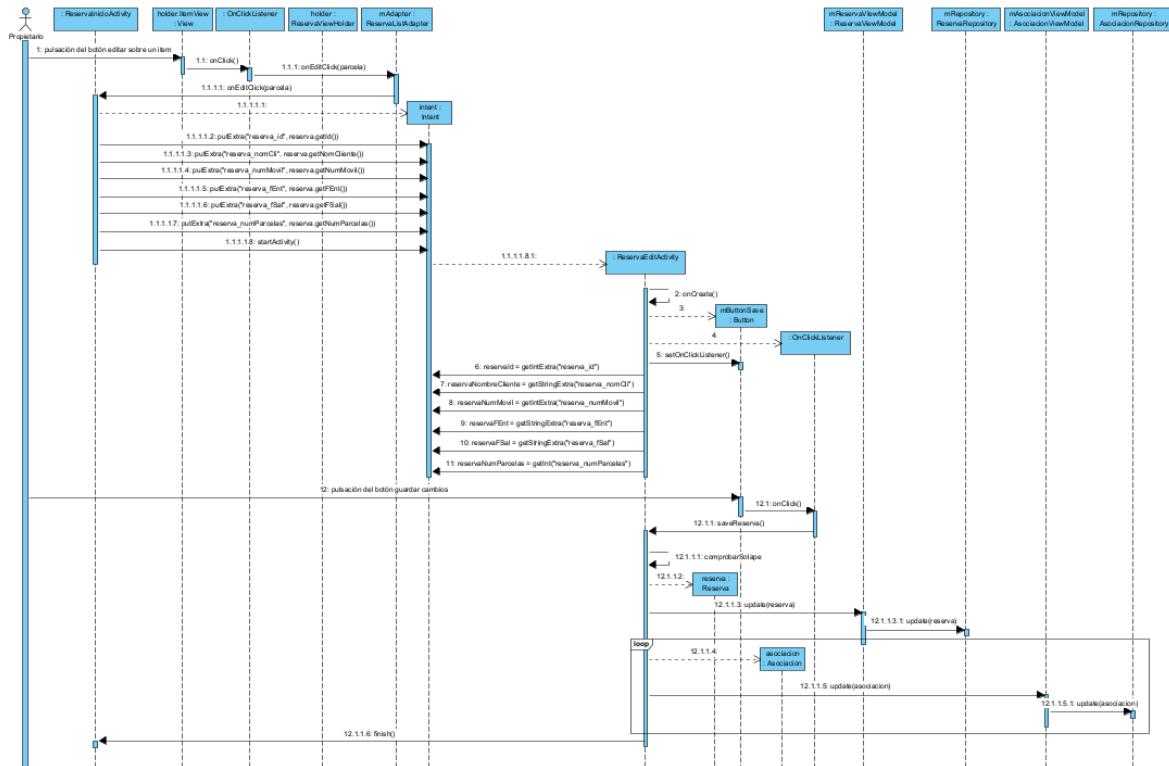
El diagrama **ReservaCreateActivity** también muestra cómo se insertaría una reserva en la base de datos, pero en este caso, también inserta en la tabla Asociación cada una de las parcelas que el propietario ha seleccionado en la reserva.



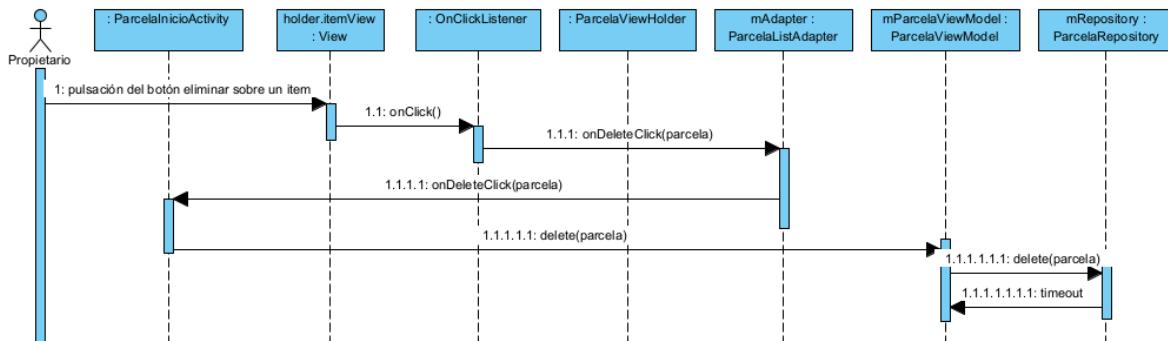
El diagrama de **ParcelaEditActivity** muestra cómo se actualizará la reserva en la base de datos si el propietario cambia alguno de los campos.



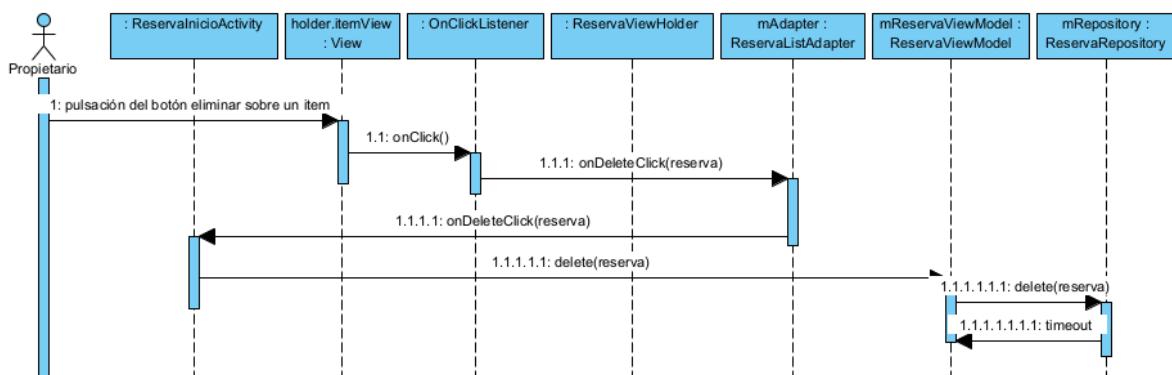
El diagrama **ReservaEditActivity** también muestra cómo se actualizará la reserva en la base de datos, incluso cómo actualizará cada parcela que la reserva tiene asociada.



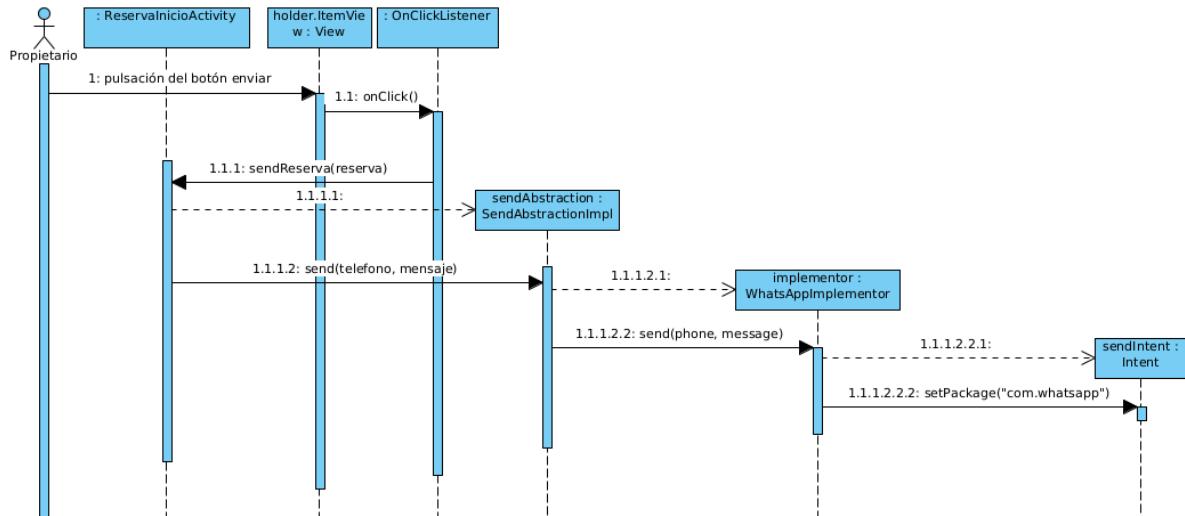
Ya por último, tenemos los diagramas de secuencia de **ParcelaEliminar**, que aunque no es una pantalla en sí de la aplicación, sí que queremos mostrar cómo se eliminaría.



El de **ReservaEliminar** también es similar, a falta de añadir la eliminación de las parcelas asociadas a la reserva.



Finalmente, el diagrama de **EnviarInformacion** muestra los pasos desde que se pulsa el botón enviar hasta que se envía el mensaje vía WhatsApp.



13. MODELO LÓGICO DE LA BASE DE DATOS RELACIONAL

Primero definimos el modelo relacional de forma gráfica, para lo que creamos 3 tablas: **PARCELA**, **RESERVA** y **ASOCIACION**. En cada tabla añadimos los atributos siguiendo lo especificado en el diagrama de clases.



Una vez diseñado el modelo, lo transformamos a SQL. Decidimos utilizar PostgreSQL, donde creamos la base de datos **ingsoft_server** y el esquema **ingsoft_schema**. Las sentencias SQL de creación de cada una de las tablas son:

```
-- Crear la tabla RESERVA
CREATE TABLE ingsoft_schema.RESERVA (
    id INTEGER PRIMARY KEY,
    nomCliente VARCHAR(255) NOT NULL,
    numMovil INTEGER NOT NULL,
    fEnt TIME(6) NOT NULL,
    fSal TIME(6) NOT NULL,
    numParcelas INTEGER NOT NULL
);
```

```
-- Crear la tabla PARCELA
```

```
CREATE TABLE ingsoft_schema.PARCELA (
    id          VARCHAR(255)   PRIMARY KEY,
    tamano      REAL           NOT NULL,
    dispAgua    BOOLEAN        NOT NULL,
    dispLuz     BOOLEAN        NOT NULL,
    numMaxOcup  INTEGER       NOT NULL,
    precioPersona REAL          NOT NULL
);
```

-- Crear la tabla ASOCIACION para la relación entre RESERVA y PARCELA

```
CREATE TABLE ingsoft_schema.ASOCIACION (
    id_reserva    INTEGER      NOT NULL,
    id_parcela    VARCHAR(255) NOT NULL,
    numOcupPorParcela  INTEGER      NOT NULL,
    PRIMARY KEY (id_reserva, id_parcela),
    FOREIGN KEY (id_reserva) REFERENCES ingsoft_schema.RESERVA(id),
    FOREIGN KEY (id_parcela) REFERENCES ingsoft_schema.PARCELA(id)
);
```

14. DISEÑO DE LA IMPLEMENTACIÓN:

Hemos comenzado la implementación de nuestra aplicación mediante la aplicación de Android Studio. Primero procedimos a analizar el ejemplo de notepad que estaba realizado sobre esta aplicación, y una vez lo entendimos correctamente, procedimos a realizar nuestra implementación.

Comenzamos con el archivo Parcelas.java, donde se encontraban definidos los atributos de la clase parcela, asignando su tipo, si eran nulos o no, la clave primaria...

A continuación, realizamos el archivo parcelaDAO.java, donde definimos las operaciones necesarias para interactuar con la base de datos camping. Estas son: insertar, actualizar, eliminar y ordenar parcelas (pudiendo ordenarlas de tres formas).

Después, hemos completado el archivo ParcelaRepository.java, el cual se encargaba de gestionar las operaciones que involucran la base de datos. Podríamos decir que este archivo es como una capa intermedia entre ParcelaDAO y las vistas. Aquí se definen los métodos declarados en el archivo DAO. Cabe destacar que utilizamos LiveData para obtener listas de parcelas en tiempo real y notificar a la interfaz del usuario si los datos cambiasen.

Respecto a la base de datos CampingRoomDatabase.java, esta almacenará la información de la clase parcela, reserva y asociación, pero de momento solo hemos implementado la clase parcela. La función de callback sirve para poblar la base de datos con datos la primera vez que se abre. Hemos declarado dos botones para poder navegar a diferentes pantallas de acuerdo con la elección del usuario y hemos implementado los métodos onCreate(), verParcelainicio() y verReservainicio(). En onCreate() se establece con setContentView que se utilizará activity_campinginicio.xml, luego se vinculan los botones definidos en el archivo .xml con los elementos del código y preparamos a los botones para ir a otras pantallas en caso de ser pulsados.

Cuando es pulsado el botón Parcelas, el usuario es llevado a otra pantalla, la cual se encuentra definida en el archivo `ParcelasListActivity.java`. Primero se inicializa el RecyclerView con el adaptador y se definen las acciones que deben hacer al editar o eliminar una parcela a través de las funciones `editParcela()` y `deleteParcela()`. En `editParcela`, se guardan todos los valores de la parcela a la que se le ha pulsado el botón, y se cambia a la pantalla `ParcelaEditActivity.java`. La función de `deleteParcela()` simplemente llama a una operación de borrado del ViewModel para que se encargue de eliminarla de la base de datos.

Seguidamente, el archivo de `ParcelaEditActivity.java` que hemos comentado en el parrafo anterior, contiene los datos que se mostrarán por pantalla (que ya han sido previamente pasados) que podrán ser modificados (excepto el identificador, debido a que es clave primaria) y se guardará así en la base de datos en cuanto se pulse guardar (actualizaremos llamando a la función de `ParcelaViewModel` de `update`).

En cuanto a las pantallas, tenemos actualmente dos: `CampingInicio.java` y `ParcelaEditActivity.java`. La primera de todas representa la pantalla principal de nuestra aplicación, donde los usuarios podrán elegir entre gestionar parcelas o reservas mediante dos botones.

El siguiente paso que realizamos fue crear el `ParcelaViewModel.java` que se encarga de gestionar el acceso al repositorio desde la interfaz de usuario (de nuevo es como una clase intermedia). Este ViewModel incluye: `getAllParcelas()`, que devuelve una lista de todas las parcelas como LiveData; `delete(Parcela parcela)`, que elimina la parcela seleccionada de la base de datos; y `update (Parcela parcela)`, que actualiza los datos de la parcela seleccionada.

Por último, `ParcelaListAdapter.java`, gestiona cómo se muestran las parcelas en el RecyclerView y controla las interacciones del usuario como editar y eliminar.

El método `onBindViewHolder()` configura cada parcela de la lista de parcelas, para que muestre su identificador de nombre y los botones de edición y eliminación.

El método `setParcelas()` permite al adapter poder actualizar la lista cada cierto tiempo.

La interfaz `OnEditClickListener` y `OnDeleteClickListener`, ayuda a preparar que al hacer click sobre cada botón, se ejecuten las acciones correspondientes que se han especificado (en este caso será ir a otra pantalla para el botón de editar, y hacer que se elimine la parcela de la base de datos para el otro botón).

15. PRUEBAS

Para acceder a la pantalla de las pruebas, en el menú principal de la aplicación debemos pulsar el botón “PRUEBAS”. Una vez dentro, tenemos 3 pruebas disponibles: las de caja negra, las de volumen y las de sobrecarga. Pulsando el botón se ejecutan las respectivas pruebas. En los logs de Android Studio podemos ver los resultados de la ejecución.



15.1. PRUEBAS DE CAJA NEGRA

15.1.1. insert de ParcelaRepository

Clases de equivalencia para el método insert de la clase ParcelaRepository

Parámetro	Clase de equivalencia válida	Clase de equivalencia no válida
parcela.id	1) parcela.id != null 2) parcela.id.length() > 0	3) parcela.id == null 4) parcela.id.length() == 0
parcela.tamano	5) parcela.tamano > 0.0	6) parcela.tamano <= 0.0
parcela.disponibilidadAgua	7) parcela.disponibilidadAgua es booleano	8) parcela.disponibilidadAgua no es booleano
parcela.disponibilidadLuz	9) parcela.disponibilidadLuz es booleano	10) parcela.disponibilidadLuz no es booleano
parcela.numMaxOcupantes	11) parcela.numMaxOcupantes	12) parcela.numMaxOcupantes

	> 0	<= 0
parcela.precioPorPersona	13) parcela.precioPorPersona > 0.0	14) parcela.precioPorPersona <= 0.0

Casos de prueba para el método insert de la clase ParcelaRepository

Caso	Parámetros	Resultado esperado	Clases cubiertas
<i>Casos de prueba para las clases de equivalencia válidas</i>			
1	parcela.id : "VALLE DE PRUEBA" parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	> 0	1, 2, 5, 7, 9, 11, 13
<i>Casos de prueba para las clases de equivalencia no válidas</i>			
2	parcela.id : null parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	-1	3
3	parcela.id : "" parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	-1	4
4	parcela.id : "VALLE DE PRUEBA" parcela.tamano : 0.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	-1	6
5	parcela.id : "VALLE DE PRUEBA" parcela.tamano : 12.0 parcela.disponibilidadAgua : 3 parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	-1	8
6	parcela.id : "VALLE DE PRUEBA" parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : 3	-1	10

	parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0		
7	parcela.id : "VALLE DE PRUEBA" parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 0 parcela.precioPorPersona : 30.0	-1	12
8	parcela.id : "VALLE DE PRUEBA" parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 0.0	-1	14

15.1.2. update de ParcelaRepository

Clases de equivalencia para el método update de la clase ParcelaRepository

Parámetro	Clase de equivalencia válida	Clase de equivalencia no válida
parcela.tamano	1) parcela.tamano > 0.0	2) parcela.tamano <= 0.0
parcela.disponibilidadAgua	3) parcela.disponibilidadAgua es booleano	4) parcela.disponibilidadAgua no es booleano
parcela.disponibilidadLuz	5) parcela.disponibilidadLuz es booleano	6) parcela.disponibilidadLuz no es booleano
parcela.numMaxOcupantes	7) parcela.numMaxOcupantes > 0	8) parcela.numMaxOcupantes <= 0
parcela.precioPorPersona	9) parcela.precioPorPersona > 0.0	10) parcela.precioPorPersona <= 0.0

Casos de prueba para el método update de la clase ParcelaRepository

Caso	Parámetros	Resultado esperado	Clases cubiertas
<i>Casos de prueba para las clases de equivalencia válidas</i>			
1	parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	1	1, 3, 5, 7, 9
<i>Casos de prueba para las clases de equivalencia no válidas</i>			
2	parcela.tamano : 0.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	0	2
3	parcela.tamano : 12.0 parcela.disponibilidadAgua : 3 parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0	0	4
4	parcela.tamano : 12.0 parcela.disponibilidadAgua : true	0	6

	parcela.disponibilidadLuz : 3 parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 30.0		
5	parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 0 parcela.precioPorPersona : 30.0	0	8
6	parcela.tamano : 12.0 parcela.disponibilidadAgua : true parcela.disponibilidadLuz : true parcela.numMaxOcupantes : 3 parcela.precioPorPersona : 0.0	0	10

15.1.3. delete de ParcelaRepository

Clases de equivalencia para el método delete de la clase ParcelaRepository

Parámetro	Clase de equivalencia válida	Clase de equivalencia no válida
parcela.id	1) parcela.id != null 2) parcela.id.length() > 0	3) parcela.id == null 4) parcela.id.length() == 0

Casos de prueba para el método delete de la clase ParcelaRepository

Caso	Parámetros	Resultado esperado	Clases cubiertas
<i>Casos de prueba para las clases de equivalencia válidas</i>			
1	parcela.id : "VALLE DE PRUEBA"	1	1, 2
<i>Casos de prueba para las clases de equivalencia no válidas</i>			
2	parcela.id : null	0	3
3	parcela.id : ""	0	4

15.1.4. insert de ReservaRepository

Clases de equivalencia para el método insert de la clase ReservaRepository

Parámetro	Clase de equivalencia válida	Clase de equivalencia no válida
reserva.id	1) reserva.id >= 0	2) reserva.id < 0
reserva.nomCliente	3) reserva.nomCliente != null 4) reserva.nomCliente.length()>0	5) reserva.nomCliente == null 6) reserva.nomCliente.length() == 0
reserva.numMovil	7) reserva.numMovil.length()==9 8) $\forall i, \text{reserva.numMovil}[i] \in [0-9]$	9)reserva.numMovil.length()<9 10)reserva.numMovil.length()>9 11) $\exists i, \text{reserva.numMovil}[i] \notin [0-9]$
reserva.fEnt	12) reserva.fEnt.length() == 10 13) $\forall i \in [0,1,3,4,6,7,8,9], \text{reserva.fEnt}[i] \in [0-9]$ 14) $\forall i \in [2,5], \text{reserva.fEnt}[i] == '/'$	15) reserva.fEnt.length() < 10 16) reserva.fEnt.length() > 10 17) $\exists i \in [0,1,3,4,6,7,8,9], \text{reserva.fEnt}[i] \notin [0-9]$ 18) $\exists i \in [2,5], \text{reserva.fEnt}[i] \neq '/'$
reserva.fSal	19) reserva.fEnt.length() == 10 20) $\forall i \in [0,1,3,4,6,7,8,9], \text{reserva.fEnt}[i] \in [0-9]$ 21) $\forall i \in [2,5], \text{reserva.fEnt}[i] == '/'$	22) reserva.fEnt.length() < 10 23) reserva.fEnt.length() > 10 24) $\exists i \in [0,1,3,4,6,7,8,9], \text{reserva.fEnt}[i] \notin [0-9]$ 25) $\exists i \in [2,5], \text{reserva.fEnt}[i] \neq '/'$
reserva.numParcelas	26) reserva.numParcelas > 0	27) reserva.numParcelas <= 0

Casos de prueba para el método insert de la clase ReservaRepository

Caso	Parámetros	Resultado esperado	Clases cubiertas
<i>Casos de prueba para las clases de equivalencia válidas</i>			
1	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	> 0	1, 3, 4, 7, 8, 12, 13, 14, 19, 20, 21, 26
<i>Casos de prueba para las clases de equivalencia no válidas</i>			
2	reserva.id : -5	-1	2

	reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2		
3	reserva.id : 1 reserva.nomCliente : null reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	5
4	reserva.id : 1 reserva.nomCliente : "" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	6
5	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 62 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	9
6	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 6622159588888 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	10
7	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662-15a58 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	11
8	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/6/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	15
9	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 0012/0006/2025 reserva.fSal : 15/06/2025	-1	16

	reserva.numParcelas : 2		
10	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/ju/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	17
11	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12-06-2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	-1	18
12	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/6/2025 reserva.numParcelas : 2	-1	22
13	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 0015/0006/2025 reserva.numParcelas : 2	-1	23
14	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/ju/2025 reserva.numParcelas : 2	-1	24
15	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15-06-2025 reserva.numParcelas : 2	-1	25
16	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 0	-1	27

15.1.5. update de ReservaRepository

Clases de equivalencia para el método update de la clase ReservaRepository

Parámetro	Clase de equivalencia válida	Clase de equivalencia no válida
reserva.nomCliente	1) reserva.nomCliente != null 2) reserva.nomCliente.length() > 0	3) reserva.nomCliente == null 4) reserva.nomCliente.length() == 0
reserva.numMovil	5) reserva.numMovil.length() == 9 6) $\forall i$, $reserva.numMovil[i] \in [0-9]$	7) reserva.numMovil.length() < 9 8) reserva.numMovil.length() > 9 9) $\exists i, reserva.numMovil[i] \notin [0-9]$
reserva.fEnt	10) reserva.fEnt.length() == 10 11) $\forall i \in [0,1,3,4,6,7,8,9]$, $reserva.fEnt[i] \in [0-9]$ 12) $\forall i \in [2,5]$, $reserva.fEnt[i] == '/'$	13) reserva.fEnt.length() < 10 14) reserva.fEnt.length() > 10 15) $\exists i \in [0,1,3,4,6,7,8,9]$, $reserva.fEnt[i] \notin [0-9]$ 16) $\exists i \in [2,5]$, $reserva.fEnt[i] \neq '/'$
reserva.fSal	17) reserva.fEnt.length() == 10 18) $\forall i \in [0,1,3,4,6,7,8,9]$, $reserva.fEnt[i] \in [0-9]$ 19) $\forall i \in [2,5]$, $reserva.fEnt[i] == '/'$	20) reserva.fEnt.length() < 10 21) reserva.fEnt.length() > 10 22) $\exists i \in [0,1,3,4,6,7,8,9]$, $reserva.fEnt[i] \notin [0-9]$ 23) $\exists i \in [2,5]$, $reserva.fEnt[i] \neq '/'$
reserva.numParcelas	24) reserva.numParcelas > 0	25) reserva.numParcelas <= 0

Casos de prueba para el método update de la clase ReservaRepository

Caso	Parámetros	Resultado esperado	Clases cubiertas
<i>Casos de prueba para las clases de equivalencia válidas</i>			
1	reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	1	1, 2, 5, 6, 10, 11, 12, 17, 18, 19, 24
<i>Casos de prueba para las clases de equivalencia no válidas</i>			
2	reserva.nomCliente : null reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025	0	3

	reserva.numParcelas : 2		
3	reserva.nomCliente : "" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	0	4
4	reserva.nomCliente : "Pepe" reserva.numMovil : 62 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	0	7
5	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 6622159588888 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	0	8
6	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662-15a58 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	0	9
7	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/6/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	0	13
8	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 0012/0006/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	0	14
9	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/ju/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 2	0	15
10	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12-06-2025 reserva.fSal : 15/06/2025	0	16

	reserva.numParcelas : 2		
11	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/6/2025 reserva.numParcelas : 2	0	20
12	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 0015/0006/2025 reserva.numParcelas : 2	0	21
13	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/ju/2025 reserva.numParcelas : 2	0	22
14	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15-06-2025 reserva.numParcelas : 2	0	23
15	reserva.id : 1 reserva.nomCliente : "Pepe" reserva.numMovil : 662215958 reserva.fEnt : 12/06/2025 reserva.fSal : 15/06/2025 reserva.numParcelas : 0	0	25

15.1.6. delete de ReservaRepository

Clases de equivalencia para el método delete de la clase ReservaRepository

Parámetro	Clase de equivalencia válida	Clase de equivalencia no válida
reserva.id	1) reserva.id >= 0	2) reserva.id < 0

Casos de prueba para el método delete de la clase ReservaRepository

Caso	Parámetros	Resultado esperado	Clases cubiertas
------	------------	--------------------	------------------

<i>Casos de prueba para las clases de equivalencia válidas</i>			
1	reserva.id : 1	1	1
<i>Casos de prueba para las clases de equivalencia no válidas</i>			
3	reserva.id : -1	0	2

15.2. PRUEBAS DE VOLUMEN

Para las pruebas de volumen, hemos probado a insertar el máximo + 1 de parcelas y reservas.

Primero, para las parcelas, como el máximo es 100, hemos comprobado las que ya hay insertadas y hemos insertado las restantes hasta llegar a 100. Luego insertamos una más que es la que nos da error y no nos deja insertar. Para volver a dejar todo como estaba, borramos de la base de datos todas las que hemos insertado para probar.

Por otro lado, con las reservas hacemos algo similar, insertamos hasta que haya 10000 reservas. Seguidamente insertamos una más que es la que nos da error. Y en este caso también borramos las reservas de prueba insertadas.

15.3. PRUEBAS DE SOBRECARGA

Para las pruebas de sobrecarga, hemos probado la longitud máxima de caracteres que puede soportar el programa, pero para no ir de uno en uno, hemos partido de una cadena de 1000 caracteres, y lo vamos duplicando iteración a iteración. En los logs podemos observar las inserciones que se van realizando correctamente hasta que falla.

16. BIBLIOGRAFÍA

Accessing data using Room DAOs. (s. f.). Android Developers.

<https://developer.android.com/training/data-storage/room/accessing-data>

How to Draw Component Diagram? (s. f.).

https://www.visual-paradigm.com/support/documents/vpuserguide/94/2581/7292_drawingcompo.html

How to Draw Deployment Diagram? (s. f.).

https://www.visual-paradigm.com/support/documents/vpuserguide/94/2582/7293_drawingdeplo.html

How to Draw Object Diagram? (s. f.).

https://www.visual-paradigm.com/support/documents/vpuserguide/94/2584/7191_drawingobjec.html

How to Draw Package Diagram? (s. f.).

https://www.visual-paradigm.com/support/documents/vpuserguide/94/2583/7192_drawingpacka.html

How to Draw Sequence Diagram? (s. f.-b).

https://www.visual-paradigm.com/support/documents/vpuserguide/94/2577/7025_drawingseque.html

How to Draw Wireframe for Android Apps? (s. f.).

<https://www.visual-paradigm.com/tutorials/android-wireframe.jsp>

How to use livedata and viewmodel with a viewholder as Lifecycle Owner? (s. f.). Stack Overflow.

<https://stackoverflow.com/questions/54825613/how-to-use-livedata-and-viewmodel-with-a-viewholder-as-lifecycle-owner>