Ingeniería del software

Diseño y desarrollo de una aplicación para la gestión de pedidos y platos de una tienda de preparación de comida

Turno de Mañanas. Grupo M34

Hecho por

JAVIER JULVE YUBERO 840710
ALEJANDRO BENEDÍ ANDRÉS 843826

Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

ÍNDICE

RESUMEN

La finalidad de las prácticas de esta asignatura consiste en la creación de una aplicación en la que el propietario de una tienda de comida para llevar pueda gestionar con mayor facilidad los platos y todos los pedidos que le hagan los clientes.

Esta aplicación para dispositivos móviles con sistema operativo Android permite al propietario de la tienda gestionar platos y pedidos. La aplicación incluye la creación, modificación y eliminación de platos, así como la gestión de pedidos con cálculo automático del precio total y envío de información al cliente.

El proceso de construcción de esta aplicación está compuesta por diferentes fases: análisis, diseño, implementación y finalmente las pruebas. A lo largo de todas las sesiones prácticas la aplicación irá entrando en las diferentes fases de construcción empezando por el análisis de los requisitos y finalizará con la realización de las pruebas para verificar el correcto funcionamiento de la aplicación.

INTRODUCCIÓN Y OBJETIVOS

En la primera práctica de la asignatura abordamos los requisitos que nuestra aplicación tiene que cumplir (fase de requisitos). Para ello se ha realizado una catálogo de requisitos, en el cual se detallan los requisitos funcionales y no funcionales de la aplicación. Además para mayor claridad se ha realizado un diagrama de casos de usos así como la descripción textual de los mismos. En esta práctica también se ha realizado el análisis de la aplicación para ello se ha establecido el modelo estático mediante diagrama de clases y el modelo dinámico mediante diagramas de secuencia.

REQUISITOS

Catálogo de requisitos

Requisitos funcionales		
Código	**Descripción**	**Prioridad**
RF-1	La aplicación debe permitir la creación de platos.	M
RF-1.1	La aplicación debe guardar de cada plato: el nombre, una descripción en texto, la categoría (Primero, Segundo o Postre) y el precio por ración.	M
RF-2	La aplicación debe permitir la consulta de platos y su ordenación por categoría, nombre o ambos criterios.	M
RF-3	La aplicación debe permitir modificar platos previamente creados.	M
RF-4	La aplicación debe permitir la eliminación de platos previamente creados.	M
RF-5	La aplicación debe permitir la creación de pedidos.	M
RF-5.1	De cada pedido el sistema debe guardar el nombre y número del cliente, la fecha y hora de la recogida del pedido, y una selección del número de raciones que desee de uno de los platos previamente creados.	M
RF-5.2	El sistema debe de tener en cuenta que la fecha y hora de recogida debe ser posterior a la fecha y hora en la cual se hizo el pedido y que solo se podrán recoger pedidos de Martes a Domingos entre las 19:30 y las 23:00.	M
RF-5.3	Cada pedido debe tener un estado asociado que inicialmente será “solicitado” pero que podrá cambiar a “preparado” o “recogido”.	M
RF-6	La aplicación debe permitir consultar un listado de pedidos creados y se podrá ordenar por nombre del cliente, número de móvil o la fecha y hora de recogida.	M
RF-6.1	El sistema debe poder filtrar los pedidos en función de su estado. (“solicitado”, “preparado” o “recogido”)	M
RF-7	La aplicación debe permitir hacer modificaciones a pedidos ya creados.	M
RF-8	La aplicación debe calcular el precio total del pedido.	M
RF-8.1	El sistema debe calcular el precio de forma automática al crear un	M

	pedido o al modificar uno existente.	
RF-8.2	Aunque el precio de un plato se modifique este no afectará al precio total de los pedidos ya realizados.	M
RF-9	La aplicación debe enviar al móvil del cliente la información completa del pedido, incluyendo el precio	M
RF-10	La aplicación debe permitir que el usuario elimine pedidos existentes.	M

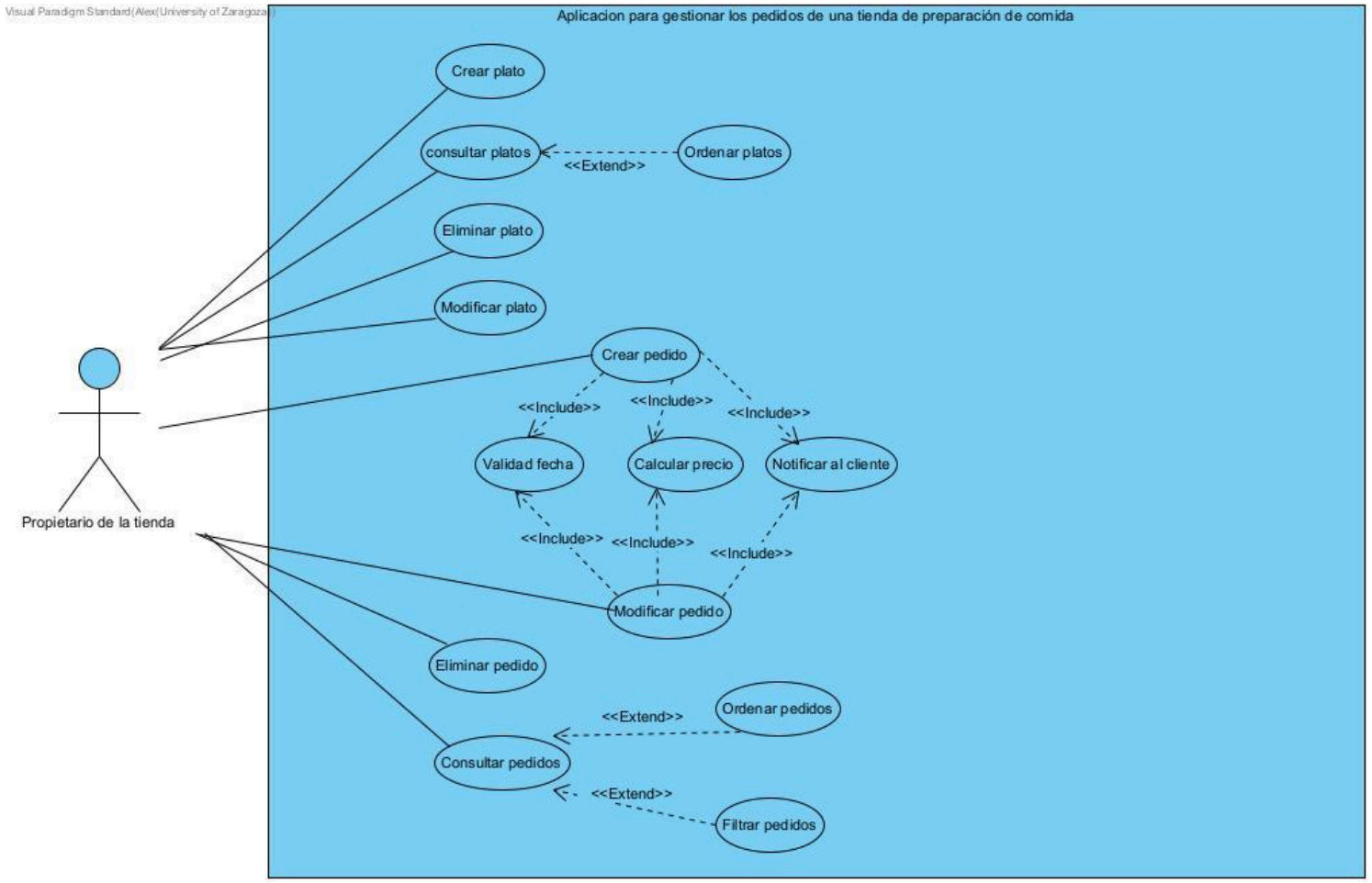
Requisitos no funcionales		
Código	**Descripción**	**Prioridad**
RNF-1	El sistema puede almacenar un número ilimitado de platos y pedidos. Sin embargo, no se espera que el sistema almacene más de 100 platos y 2000 pedidos.	M

Restricciones	
R1	La aplicación debe funcionar en dispositivos móviles con sistemas operativos android

Diagrama de casos de uso y descripción textual

Una vez establecidos los requisitos funcionales y no funcionales, realizamos el diagrama de casos de usos y la descripción textual de estos mismos para facilitar el análisis de los objetivos finales de la aplicación.

Diagrama de casos de usos



Descripción textual de los casos de usos

Crear plato

- Actor: Propietario de la tienda
- Flujo de eventos principal
    - 1. El caso de uso comienza al seleccionar la opción de crear un nuevo plato.
    - 2. El sistema muestra por pantalla el menú de creación de plato, en el cual se indican todos los datos que se tienen que introducir.
    - 3. El propietario de la tienda introduce los datos.
    - 4. El caso de uso termina cuando el propietario de la tienda confirma la creación del plato pulsando el botón de aceptar.
- Flujo de eventos alternativos:
    - 5. El caso de uso comienza cuando el propietario pulsa el botón de aceptar y faltan datos por introducir.
    - 6. El sistema muestra un mensaje de error y se reinicia el caso de uso
- Flujo de eventos alternativos
    - 7. El caso de uso comienza cuando el propietario pulsar el botón de cancelar
    - 8. El sistema cierra el menu de creacion de plato

Consultar platos

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al abrir la aplicación y selecciona la opción “platos”
    - 2. El sistema muestra por pantalla todos los platos creados anteriormente, junto con la opción de ordenar platos
    - 3. Extend: Ordenar platos
    - 4. El caso de uso termina cuando el propietario decide cerrar la aplicación o cuando se selecciona alguna otra opción

Eliminar plato

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al pulsar el botón de eliminar
    - 2. El sistema borra el plato y muestra un mensaje confirmando que el plato ha sido borrado, junto con un botón de deshacer acción
    - 3. El caso de uso termina cuando, tras un tiempo prudencial, el mensaje y el boton de deshacer acción desaparecen
- Flujo de eventos alternativo 1:
    - 1. El caso de uso comienzo cuando se pulsa el botón de deshacer acción
    - 2- El sistema restaura el último plato eliminado

Modificar plato

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al pulsar sobre el plato que se quiera modificar
    - 2. El sistema abre el menú de modificación de plato
    - 3. El propietario realizar las modificaciones oportunas
    - 4. Extend: Eliminar plato
    - 5. El caso de uso termina cuando se pulsa el botón de guardar
- Flujo de eventos alternativos:
    - 1. El caso de uso comienza cuando se pulsa el botón de cancelar
    - 2. El sistema cierra el menú de modificación de plato

Ordenar platos

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso empieza al seleccionar la opción de ordenar
    - 2. El sistema despliega un menú en el cual se encuentran todos los criterios disponibles para ordenar
    - 3. Seleccionar el criterio deseado
    - 4. El caso de uso termina cuando el sistema muestra los platos ordenados según el criterio seleccionado

Crear pedido

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al seleccionar la opción de crear pedido
    - 2. El sistema muestra por pantalla el menu el menu de creacion de pedido, junto con todos los datos que se tienen que introducir
    - 3. Se introducen los datos
    - 4. Include: Calcular precio
    - 5. Se confirma la creación del pedido, pulsado el botón de aceptar
    - 6. Include: notificar al cliente
- Flujo de eventos alternativo 1:
    - 1. En caso de uso comienza cuando se pulsa el botón de aceptar y faltan datos por introducir.
    - 2. El sistema muestra un mensaje de error y se reinicia el caso de uso
- Flujo de eventos alternativo 2:
    - 1. El caso de uso comienza cuando se pulsa el botón de cancelar
    - 2. El sistema cierra el menu de creacion de plato
- Flujo de eventos alternativo 3:
    - 1. El caso de uso comienza cuando se pulsa el botón de aceptar, pero la fecha y/o hora de recogida no es válida
    - 2. El sistema muestra un mensaje de error y se reinicia el caso de uso

Consultar pedidos

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al abrir la aplicación y selecciona la opción “pedidos”
    - 2. El sistema muestra por pantalla todos los pedidos creados anteriormente, junto con la opción de ordenar pedidos y filtrar pedidos
    - 3. Extend: Ordenar pedidos
    - 4. Extend: Filtrar pedidos
    - 5. El caso de uso termina se decide cerrar la aplicación o cuando se selecciona alguna otra opción

Eliminar pedido

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al pulsar el botón de eliminar
    - 2. El sistema borra el plato y muestra un mensaje confirmando que el pedido ha sido borrado, junto con un botón de deshacer acción
    - 3. El caso de uso termina cuando, tras un tiempo prudencial, el mensaje y el boton de deshacer acción desaparecen
- Flujo de eventos alternativo 1:
    - 1. El caso de uso comienzo cuando se pulsa el botón de deshacer acción
    - 2. El sistema restaura el último pedido eliminado

Modificar pedido

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al pulsar sobre el pedido que se quiera modificar
    - 2. El sistema abre el menú de modificación de pedido
    - 3. El propietario realizar las modificaciones oportunas
    - 4. Extend: Eliminar pedido
    - 5. Include: calcular precio
    - 6. Se guardan los cambios pulsando el botón de guardar
    - 7. Include: notificar al cliente
- Flujo de eventos alternativo 1:
    - 1. El caso de uso comienza cuando se pulsa el botón de cancelar
    - 2. El sistema cierra el menú de modificación de pedido

Ordenar pedidos

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso empieza al seleccionar la opción de ordenar
    - 2. El sistema despliega un menú en el cual se encuentran todos los criterios disponibles para ordenar
    - 3. Seleccionar el criterio deseado
    - 4. El caso de uso termina cuando el sistema muestra los pedidos ordenados según el criterio seleccionado

Filtrar pedidos

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al seleccionar la opción de filtrar
    - 2. El sistema despliega un menú en el cual se encuentran todos los criterios disponibles para filtrar
    - 3. Se selecciona el criterio deseado
    - 4. El caso de uso termina cuando el sistema muestra todos los pedidos que cumplen el criterio de filtrado

Calcular precio

- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El caso de uso comienza al introducir platos a un pedido
    - 2. Cada vez que se vayan introduciendo o eliminando platos el precio del pedido se irá actualizando
    - 3. El caso de uso termina mostrando el precio hasta ese momento del pedido

Notificar al cliente

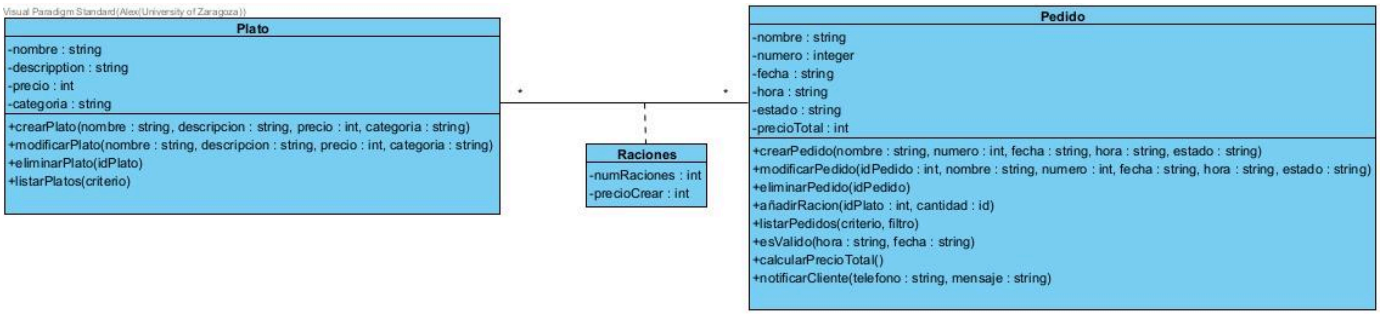
- Actor: Propietario de la tienda
- Flujo de eventos principal:
    - 1. El sistema envía al cliente, una notificación con el pedido realizado y el precio total de este

ANÁLISIS

Como ya se han realizado los requisitos de la aplicación y los casos de uso de la misma, ahora vamos a crear un diagrama de clase, también llamado modelo estático y un diagrama de secuencia o modelo dinámico.

Modelo estático

De la manera que hemos construido todo tenemos 2 clases bien diferenciadas siendo el pedido y el plato, las cuales se piden en el propio guión. Para juntar las clases tenemos una relación A continuación una descripción de cada una.



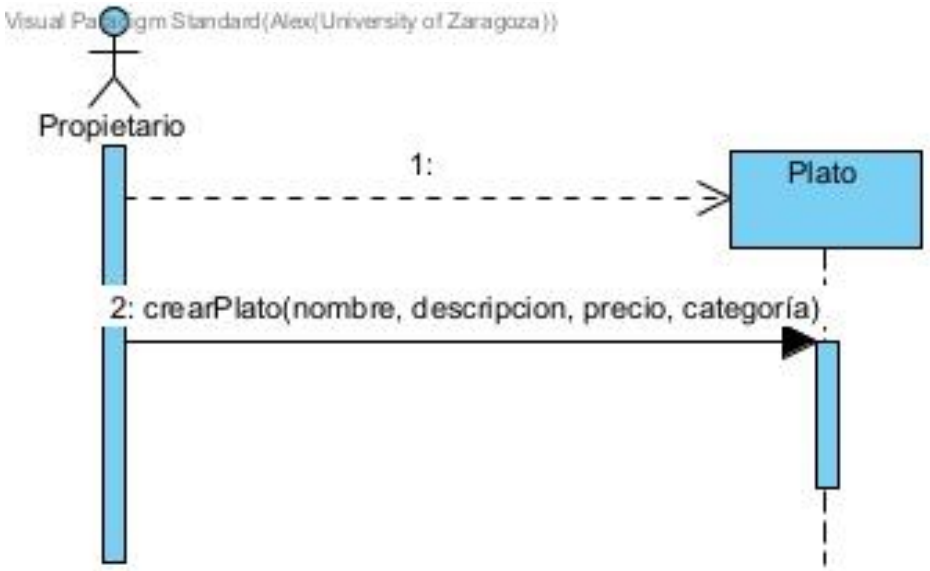
- Pedido: Representa el encargo realizado por un cliente en el cual este deberá poner su nombre, número de móvil, fecha y hora de recogida del pedido..
- Plato: Representa un plato creado en la aplicación en el que vendrá su nombre, descripción, fecha y categoria

Modelo dinámico

Para la elaboración de este modelo se ha hecho uso del modelo anterior y del diagrama de casos de uso, además se ha hecho uso de tres ventanas que indican cierto funcionamiento, siendo loop, la que indica desde dónde hasta donde se hace un bucle. opt, la cual sirve para valorar si la variable a la que va dirigida cumple su función y en caso afirmativo ejecutará ciertas acciones y en caso negativo, nada pasará. Por último tenemos alta que es igual que la anterior, solo que si no se cumple se ejecutará otra acción en lugar de no hacer nada.

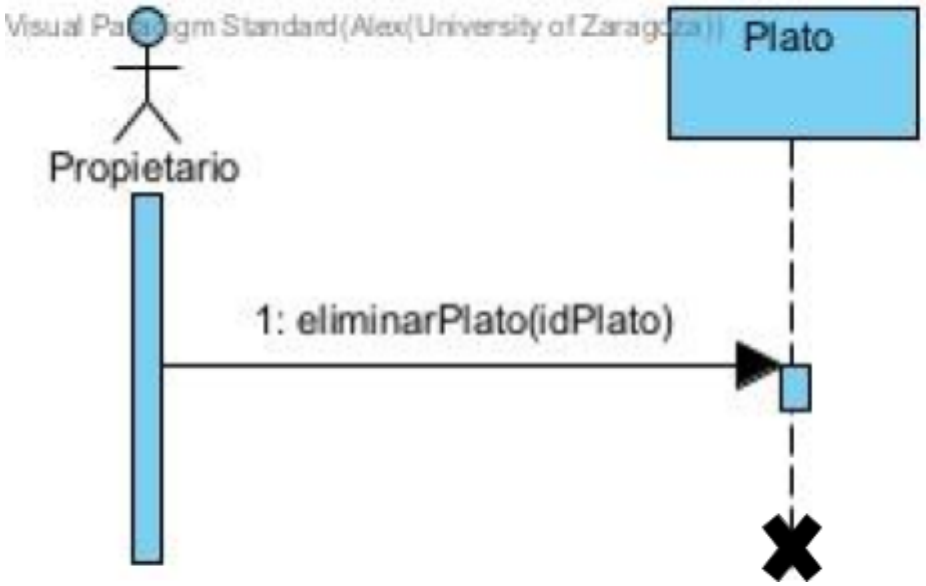
Crear plato

El propietario tiene que introducir el nombre, descripción, precio y categoría del plato.



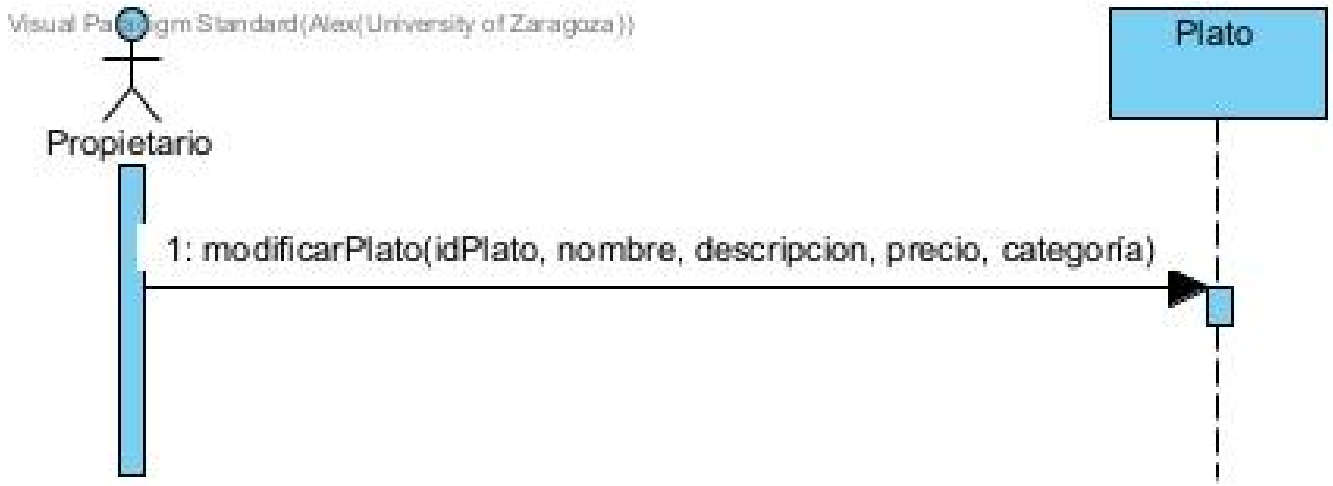
Eliminar plato

El propietario tiene que indicar que plato desea eliminar



Modificar plato

En este diagrama, el propietario le tiene que escribir las modificaciones que vea oportunas.Si no se modifica se volver a insertar los valores antiguos

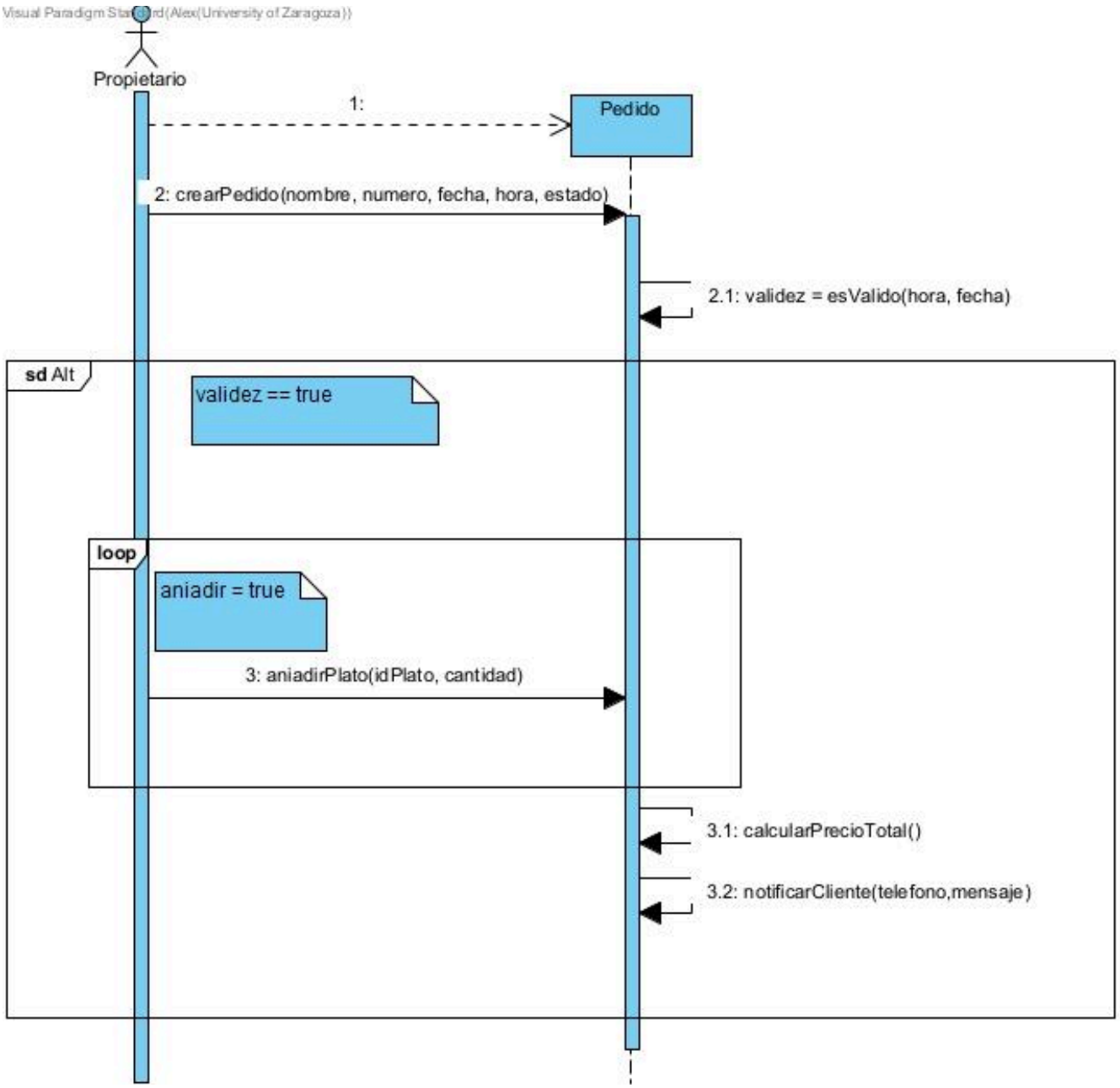


Listar Platos

En este diagrama, el propietario tiene que indicar que criterio de orden necesita. Por defecto se ordena por nombre.

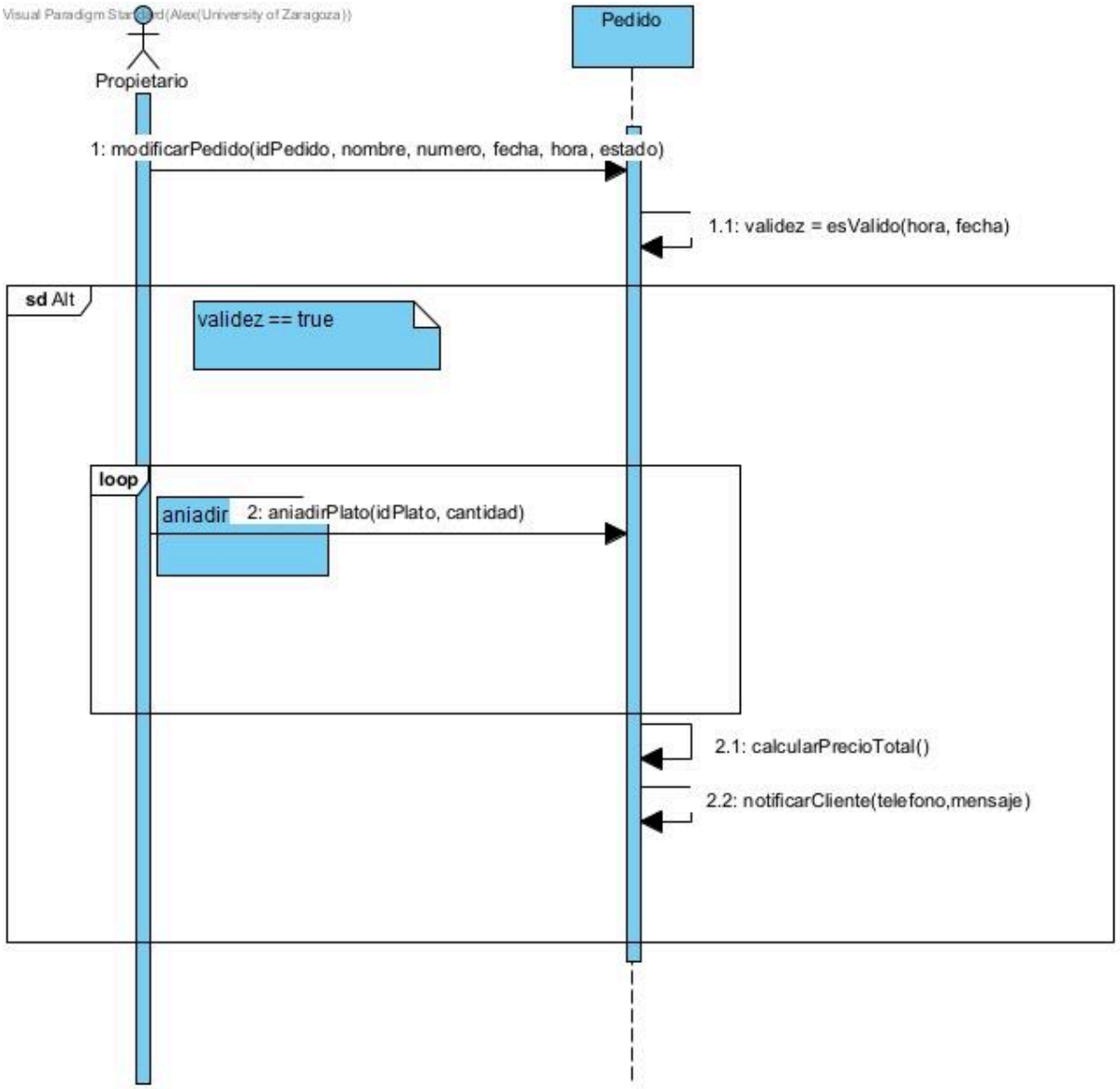
Crear pedido

Inicialmente el propietario tiene que introducir nombre, teléfono, fecha, hora y estado del pedido. Una vez introducidos, el sistema comprueba si es un pedido Si es valido se añaden los platos del pedido y las raciones asociadas a este. Una vez introducidos todos los platos se calcula el precio total y se notifica al cliente del pedido.



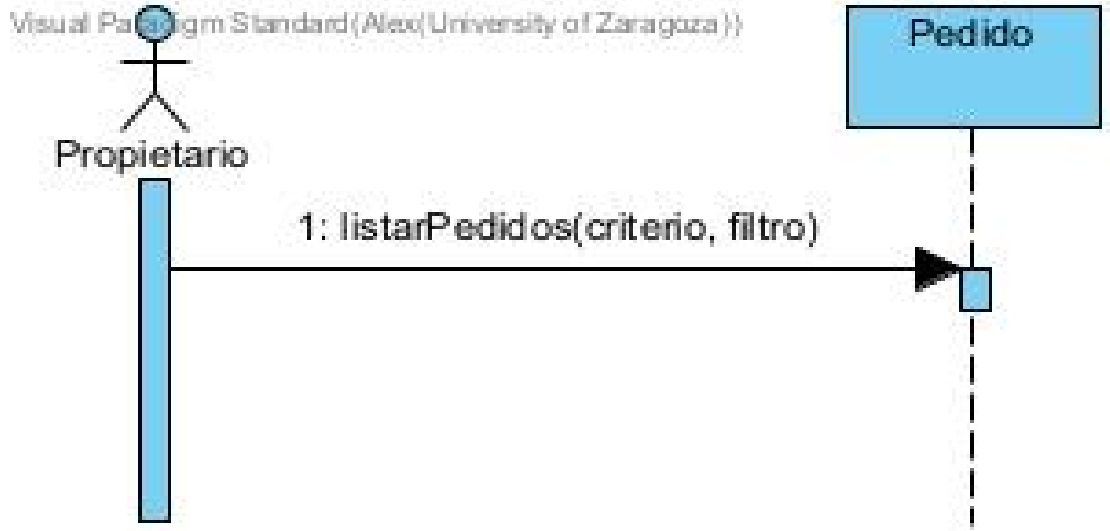
Modificar pedido

El propietario podrá introducir nombre, teléfono, fecha, hora o estado del pedido.Si no se introduce nada el sistema volverá a guardar los valores anteriores. Posteriormente y una vez comprobado que el pedido es válido, el propietario puede añadir o quitar el plato. Finalmente se calcula el precio del pedido y se notifica al cliente.



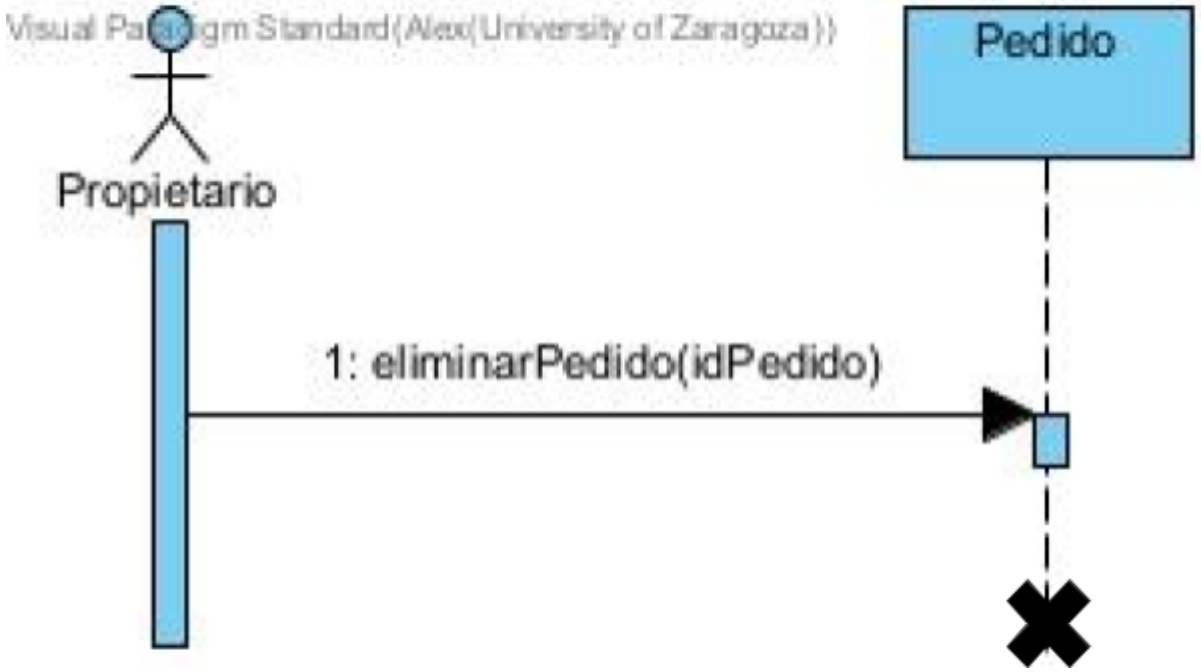
Listar pedidos

El propietario indica el criterio de orden y si quiere filtrar o no. Por defecto se ordena por nombre y no se aplica ningún criterio



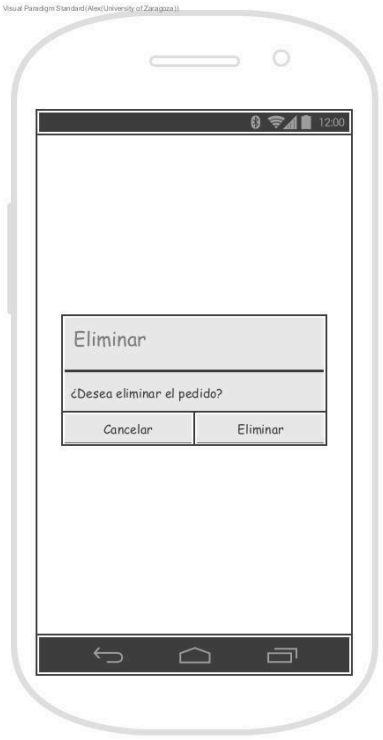
Eliminar pedido

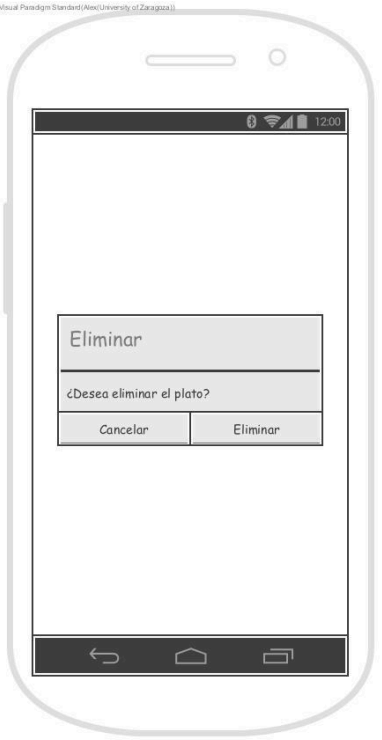
El propietario indica que pedido se quiere eliminar



PROTOTIPADO DE LA INTERFAZ

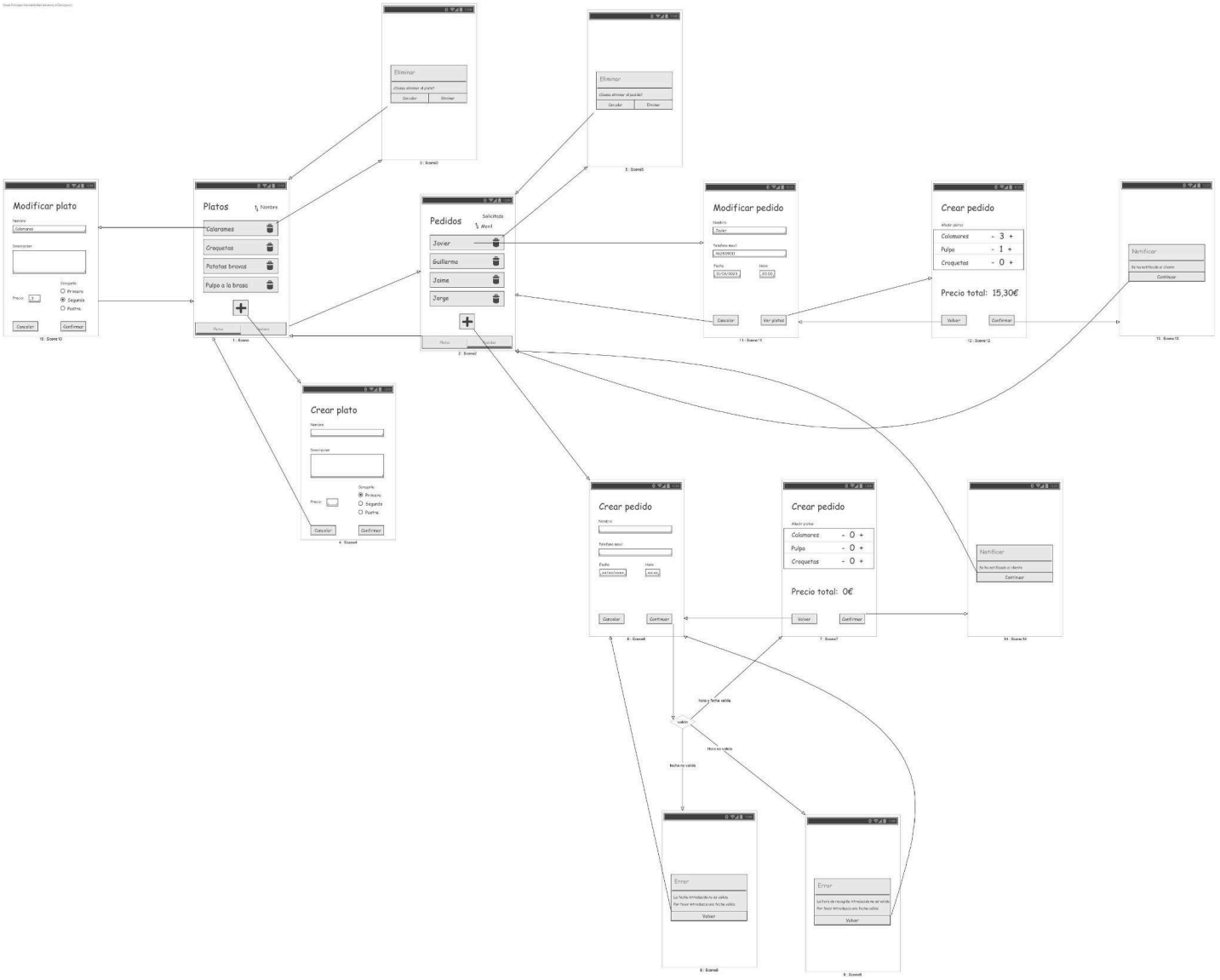
Pantallas del prototipo





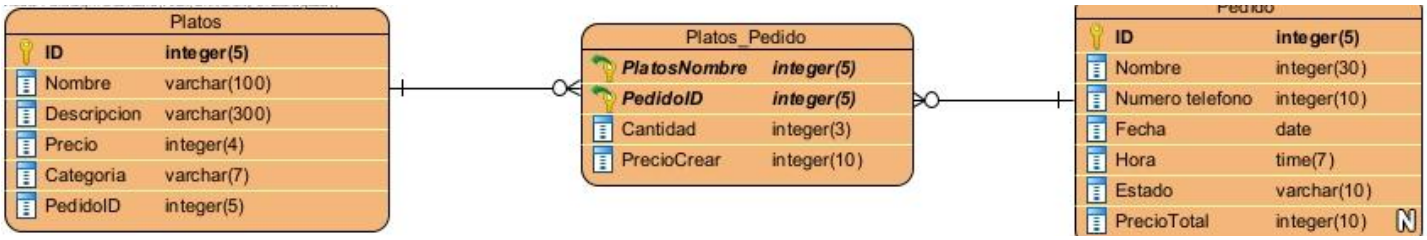
Los diálogos pasan a ser “Toast” en la implementación, excepto el de eliminar que es una actividad. El dibujo de las papeleras también ha desaparecido para eliminar y editar mantener pulsado e eligen la opción deseada

Mapas de navegación



DISEÑO DEL SISTEMA Y DE OBJETOS

Diseño lógico



Sentencias SQL

```
CREATE TABLE PLATOS (
        ID integer(4)  PRIMARY KEY,
        Descripcion varchar (300) not null,
        Precio integer(4)  not null,
        Categoria varchar(7) not null
);

CREATE TABLE PEDIDOS (
        ID integer(4)  PRIMARY KEY,
        Nombre integer(30) not null,,
        Telefono integer(10) not null,
        Fecha date not null,
        Hora time not null,
        Estado varchar(10) not null,
        PrecioTotal(10) not null
);

CREATE TABLE PLATOS_PEDIDOS (
        PlatosNombre varchar(30) REFERENCES PLATOS(Nombre),
        PedidosID integer(5) REFERENCES PEDIDOS(ID),
        Cantidad integer(3) not null,
        PrecioCrear integer(3) not null,

CONSTRAINT PK_PLATOS_PEDIDOS PRIMARY KEY (PlatosNombre, PedidosID)
);
```

Diagrama de clases a nivel de diseño

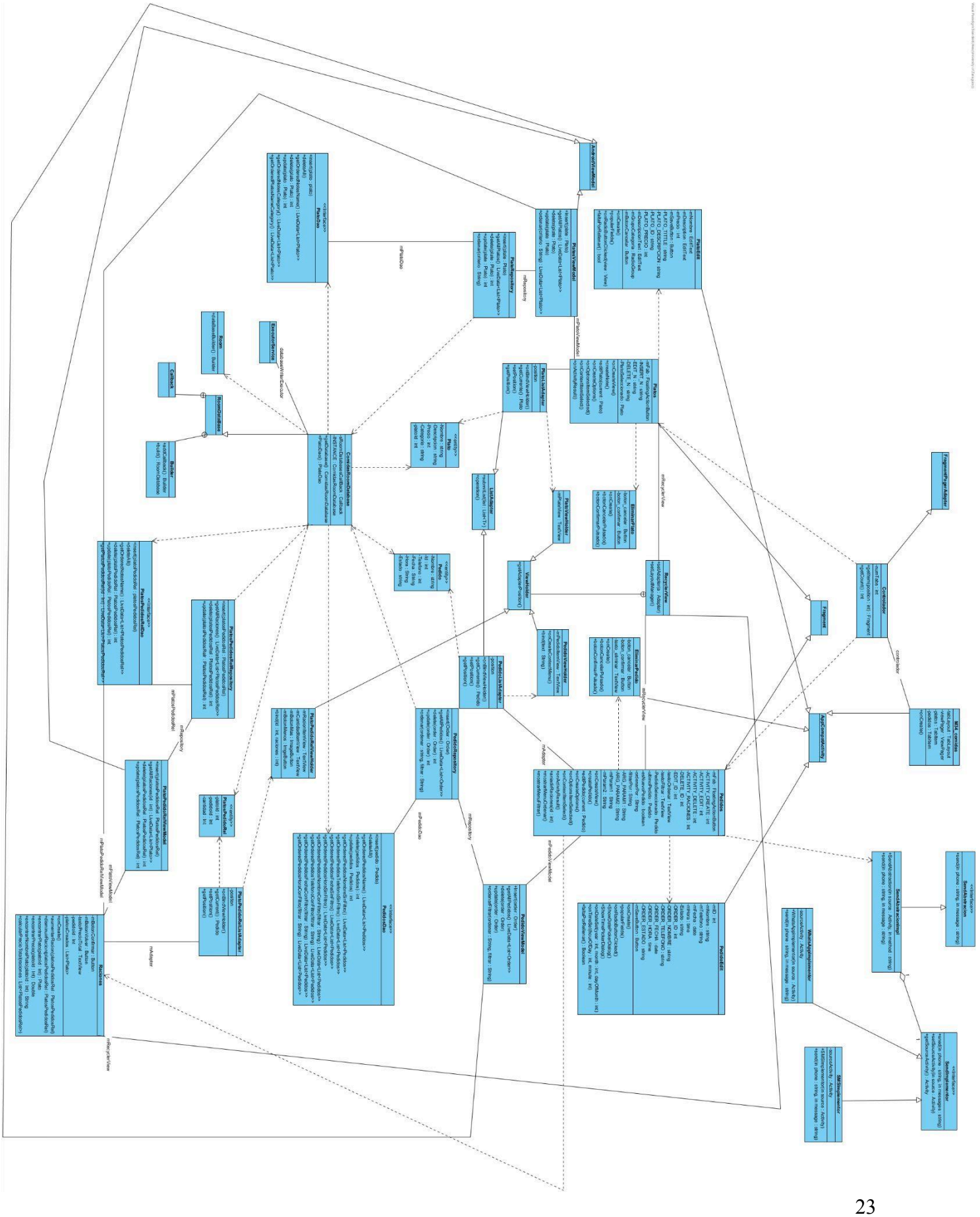
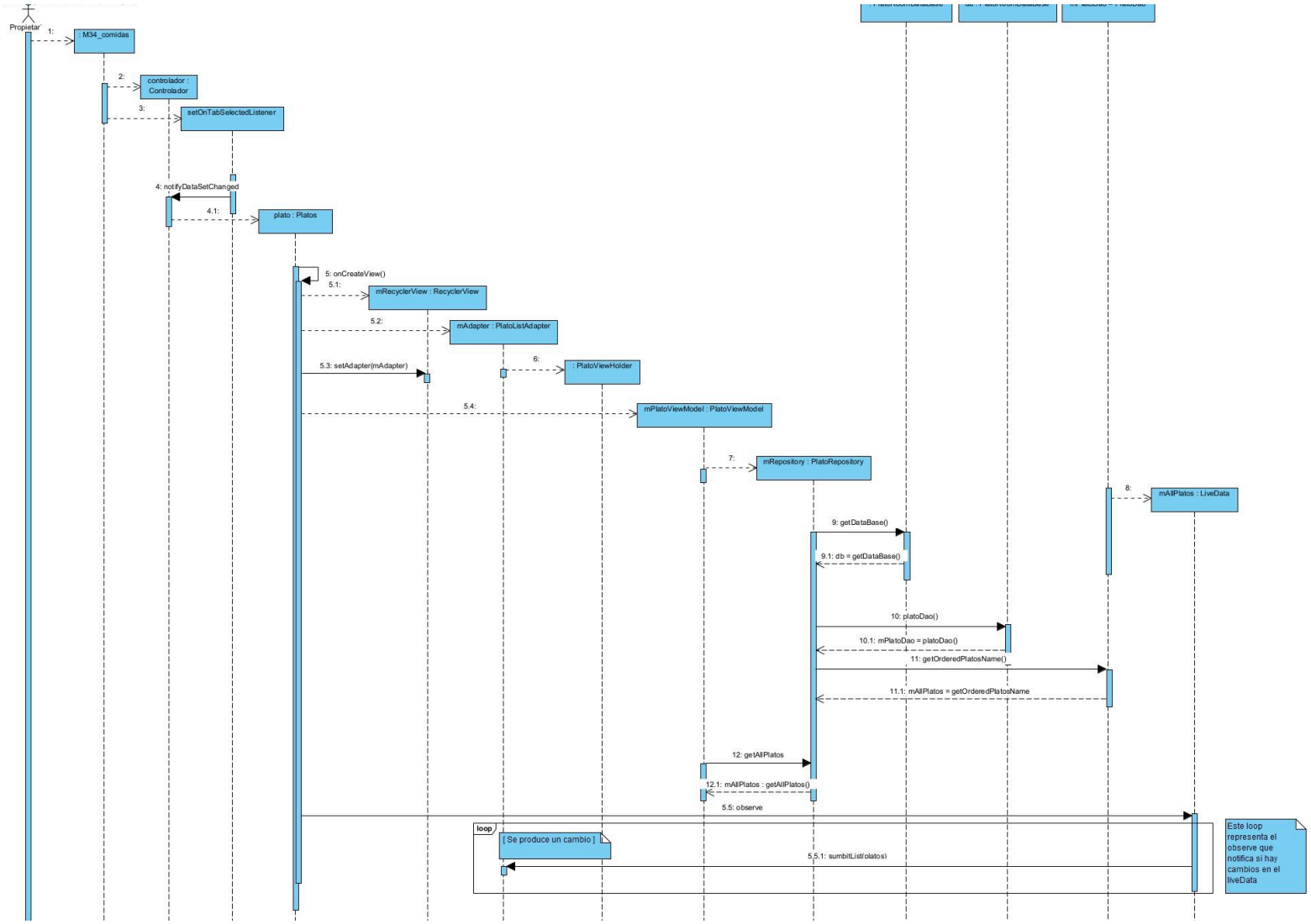


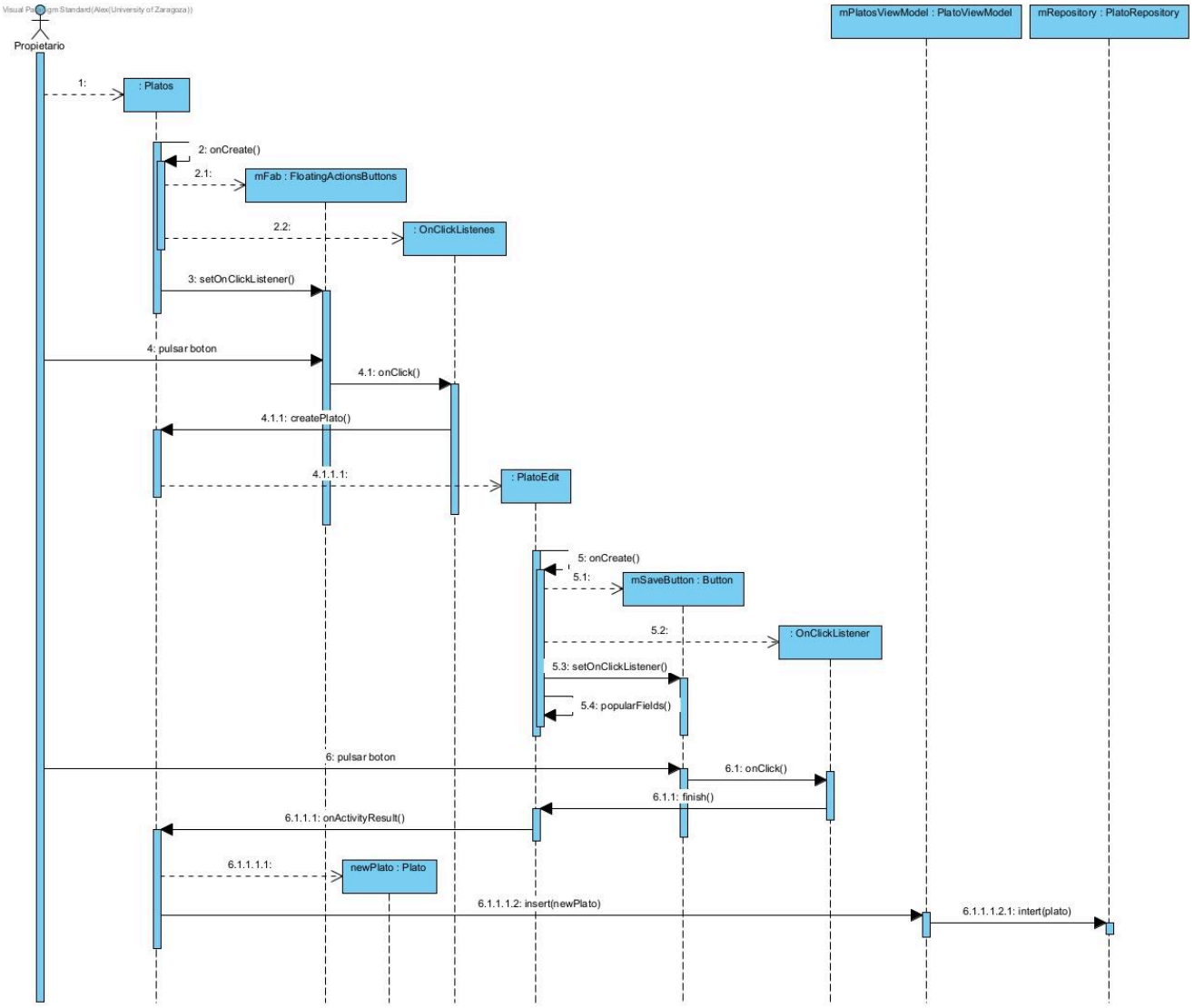
Diagrama de secuencias

Inicialización y muestra de todos los platos

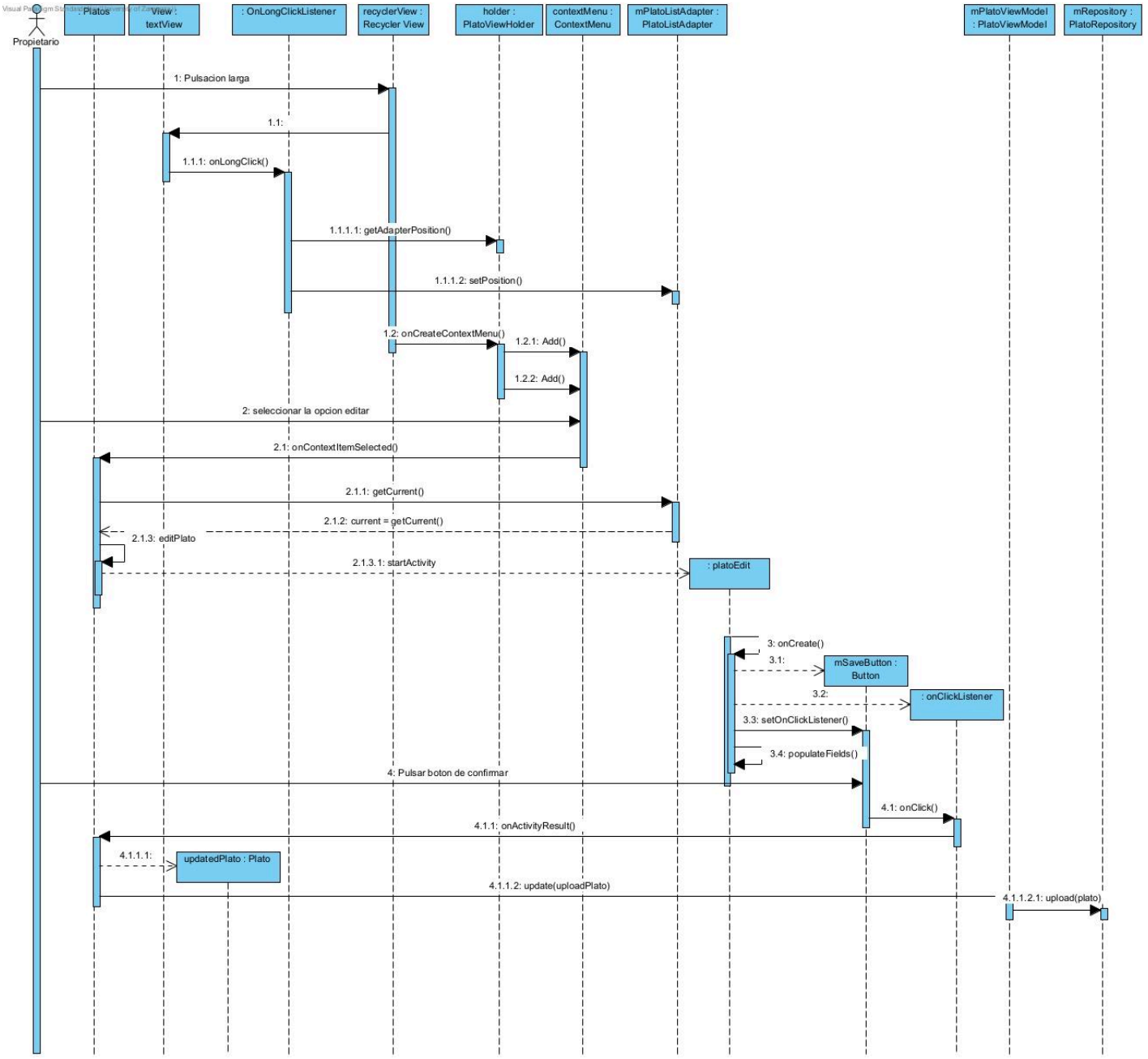


Este diagrama representa todo el proceso de inicialización del fragmento y la visualización de la lista con todos los platos. Una vez inicializado tan solo se ejecutara el observe para comprobar si se ha modificado, si es así notifica y ejecuta lo que tiene en su interior.

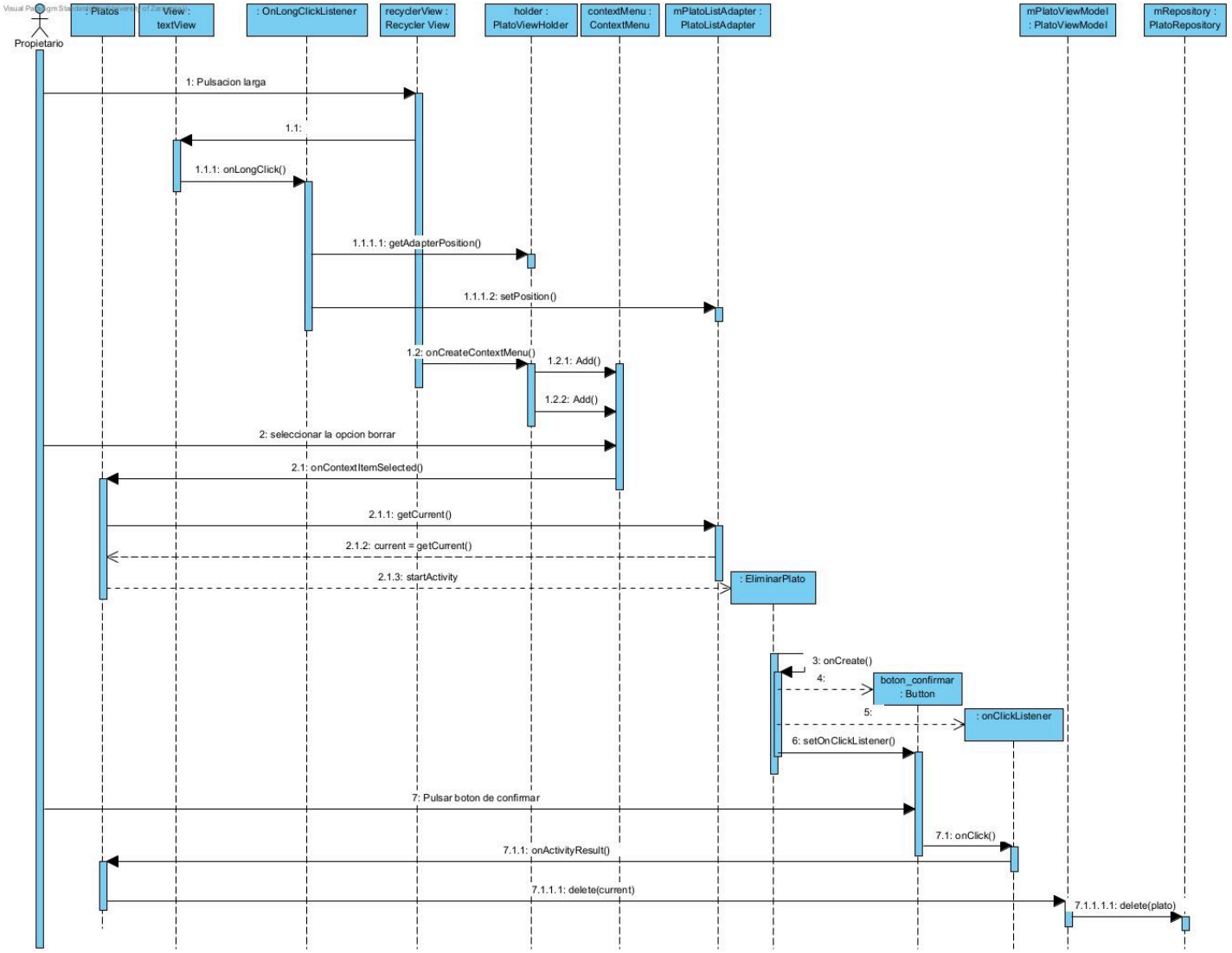
Añadir plato



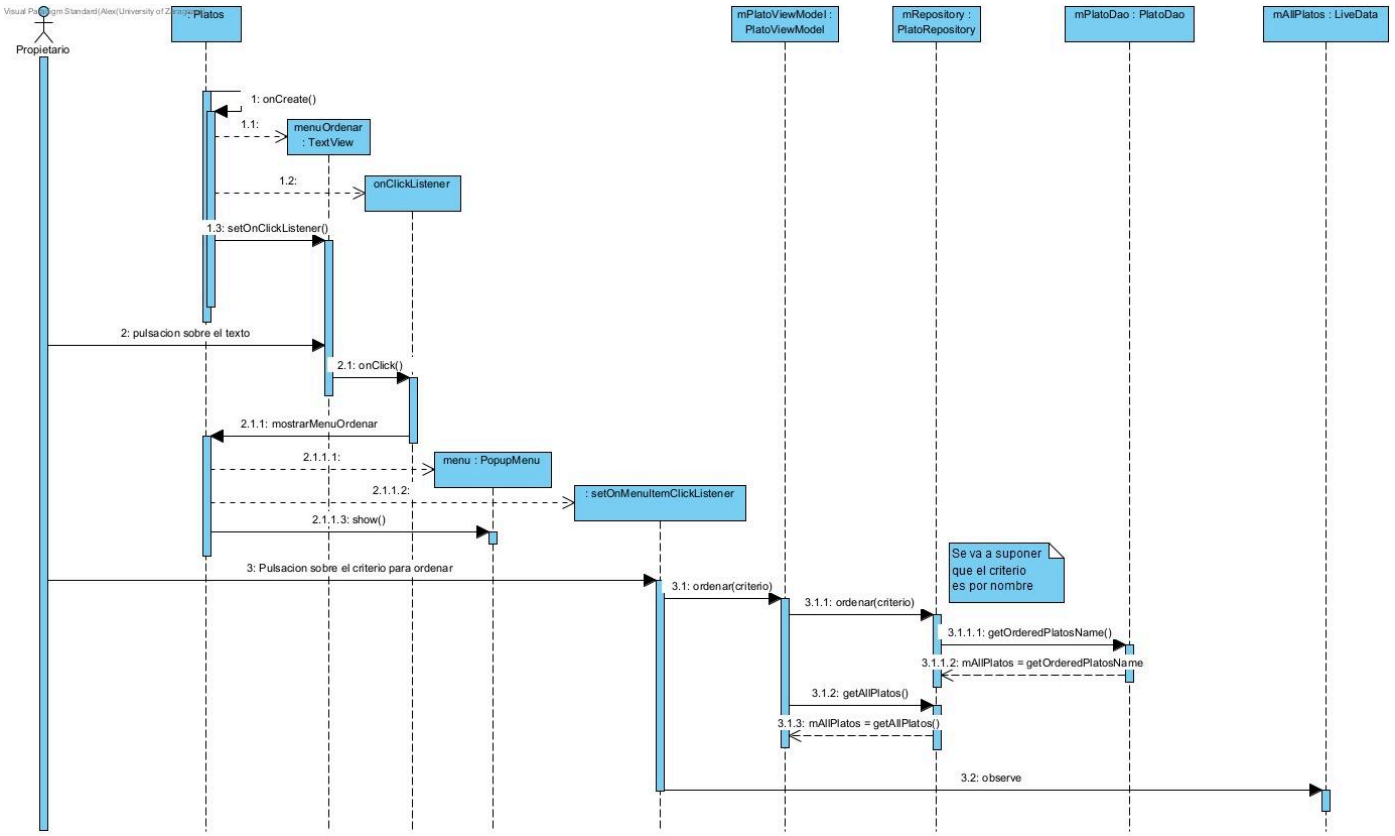
Editar plato



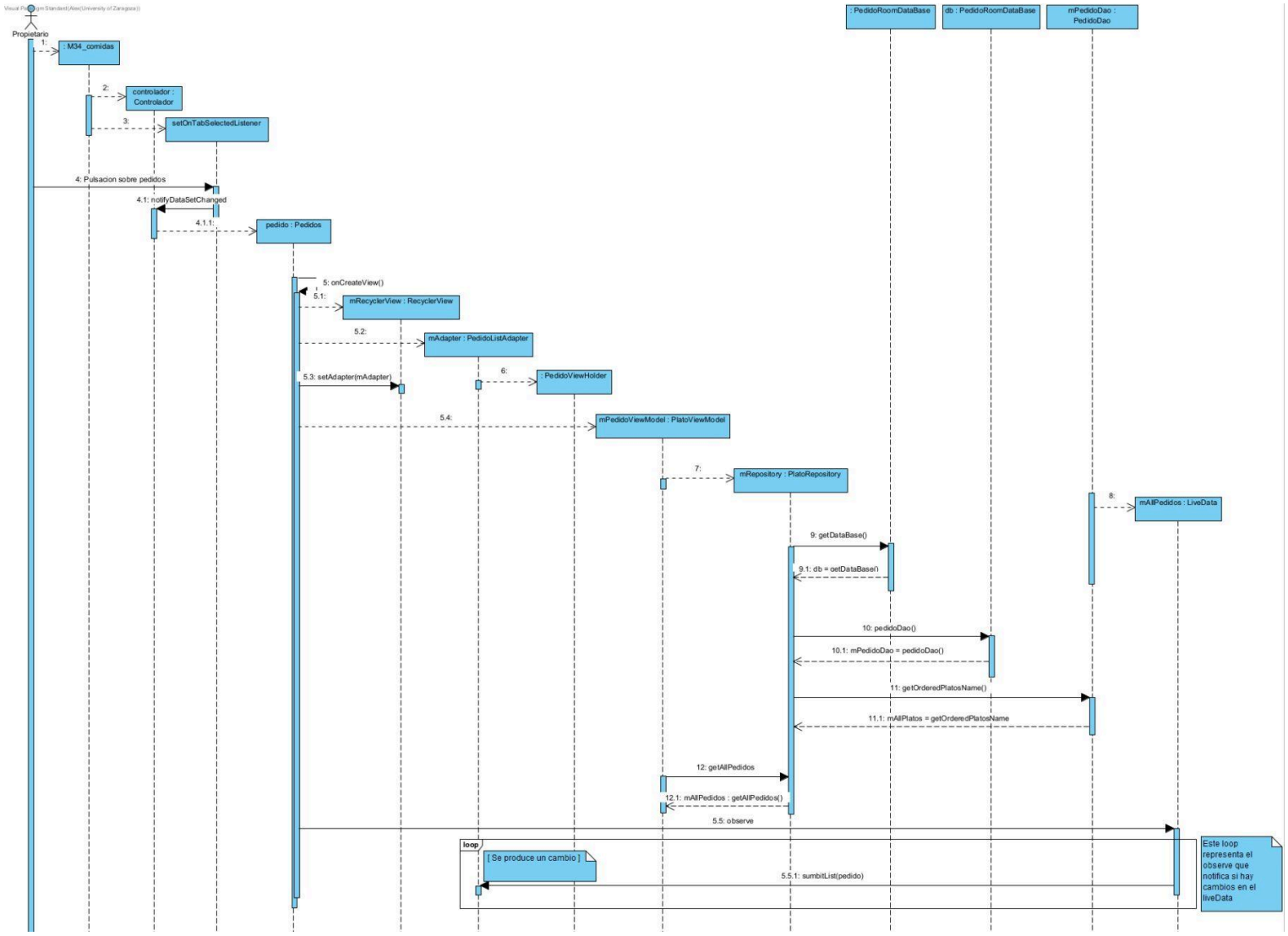
Borrar plato



Ordenar platos

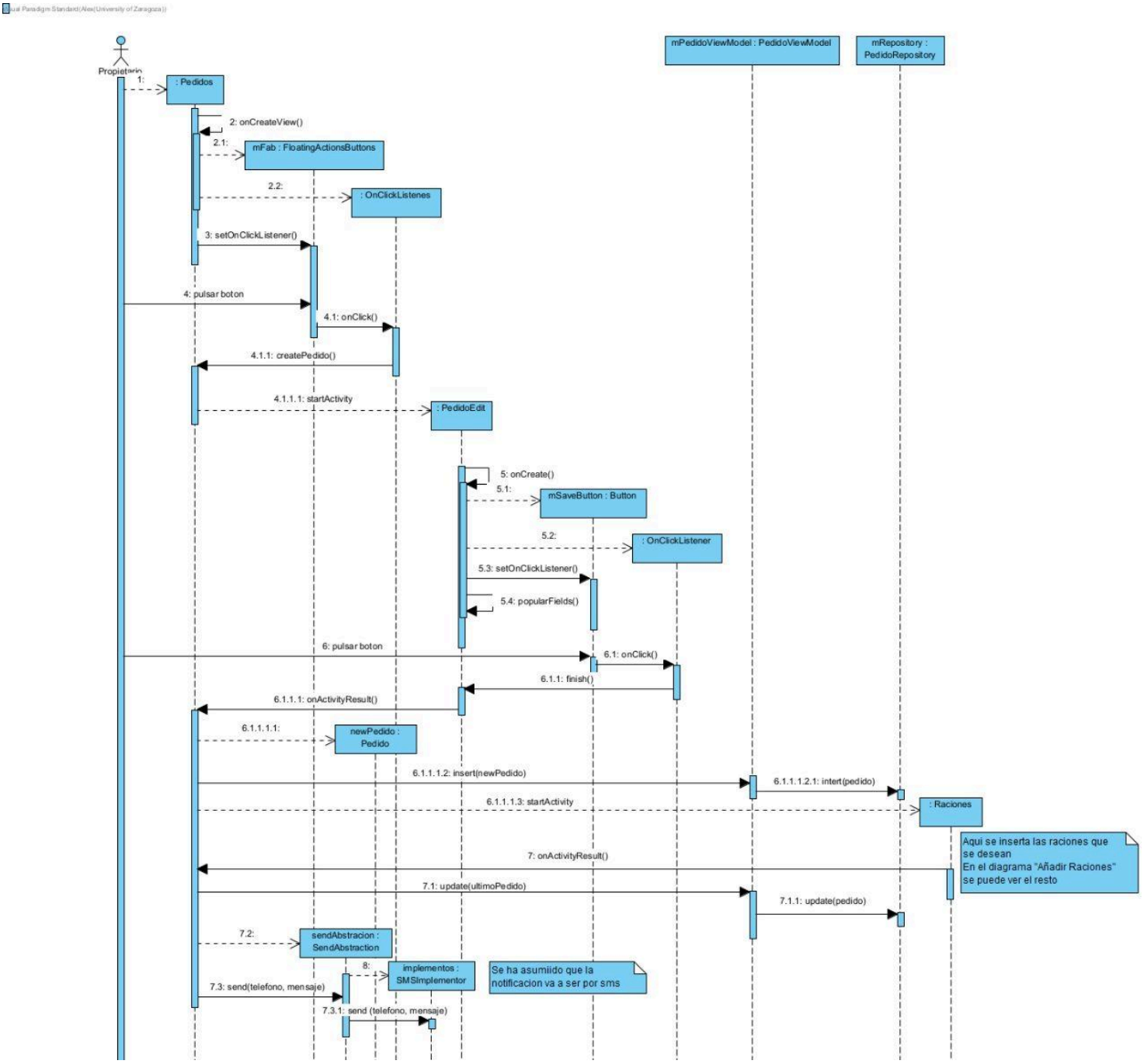


Inicializacion y muestra todos los pedidos



Este diagrama representa todo el proceso de inicialización del fragmento y la visualización de la lista con todos los pedidos. Una vez inicializado tan solo se ejecutara el observe para comprobar si se ha modificado, si es así notifica y ejecuta lo que tiene en su interior.

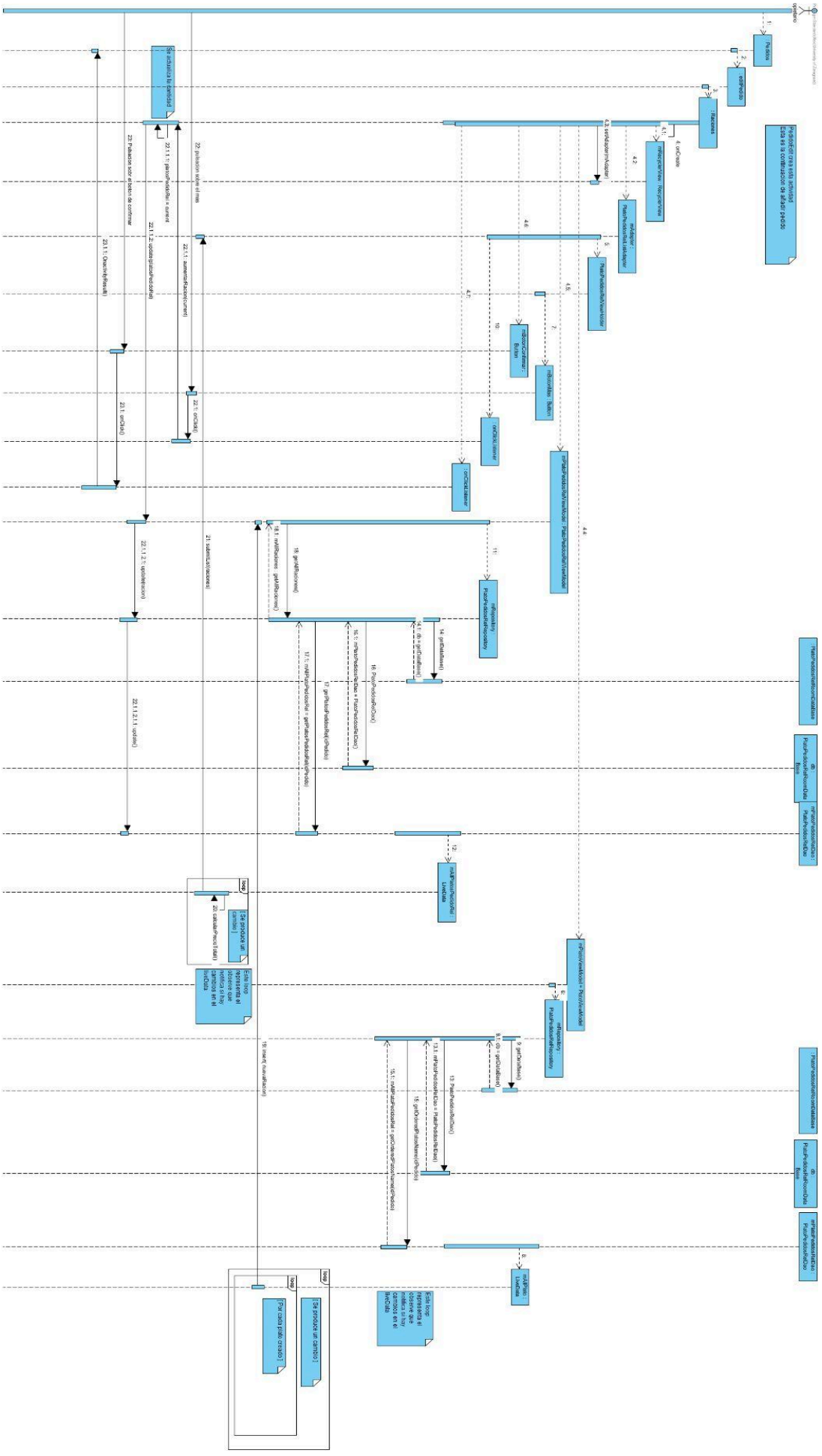
Añadir pedido



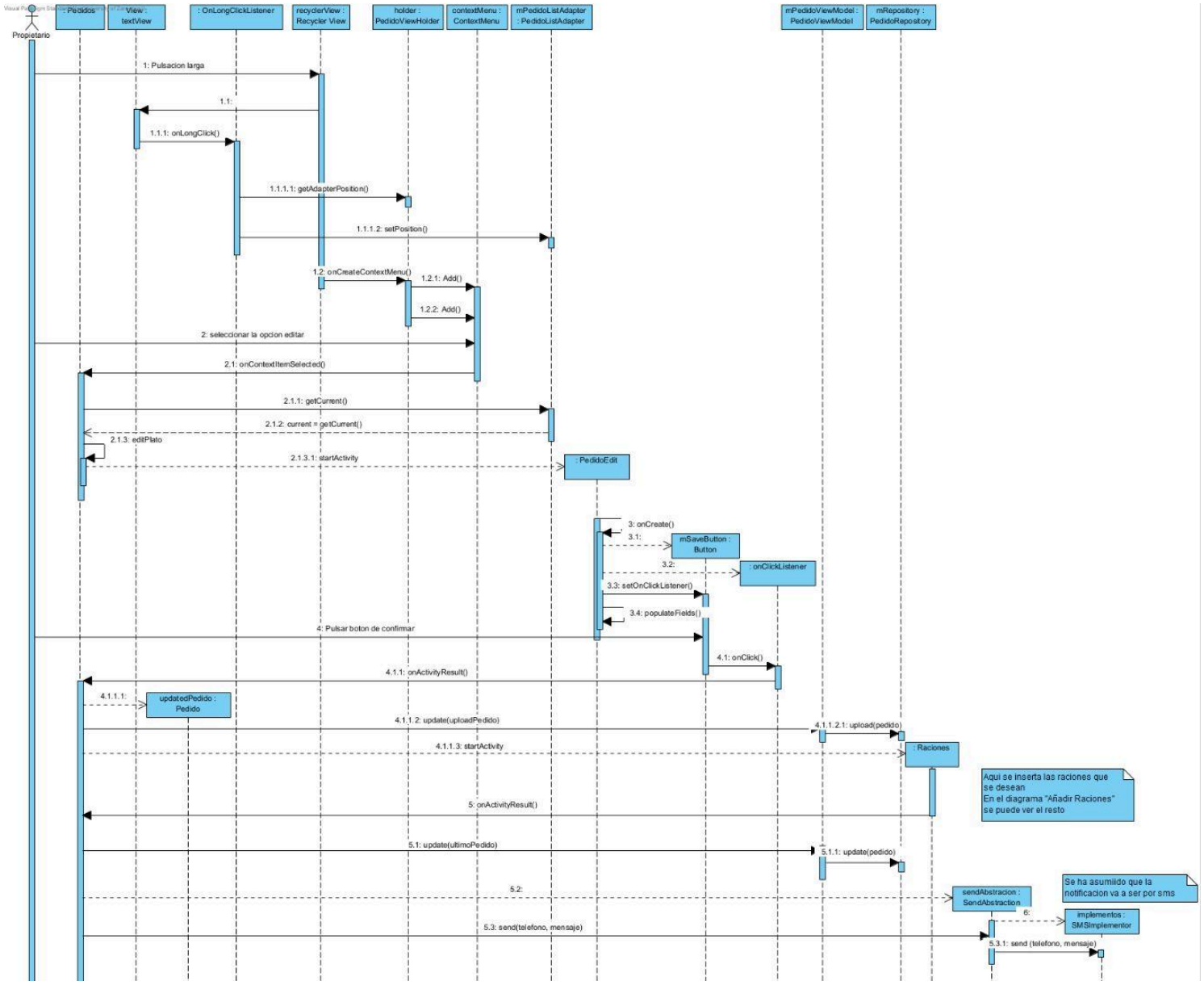
Al añadir el pedido también se añaden las raciones. Por motivos de espacio lo he separó en 2 diagramas. Este diagrama unicamente añade en los campos “Nombre”, “Telefono”, “Fecha”, “Hora”. Una vez añadido esto en la base de datos se pasa a la actividad Raciones. En ella se añaden todas las raciones deseadas. En la siguiente pagina se puede ver su diagrama de estados

Tras la ejecución de la actividad de raciones se vuelve de nuevo a la actividad Pedido para notificar al cliente por sms

Añadir raciones

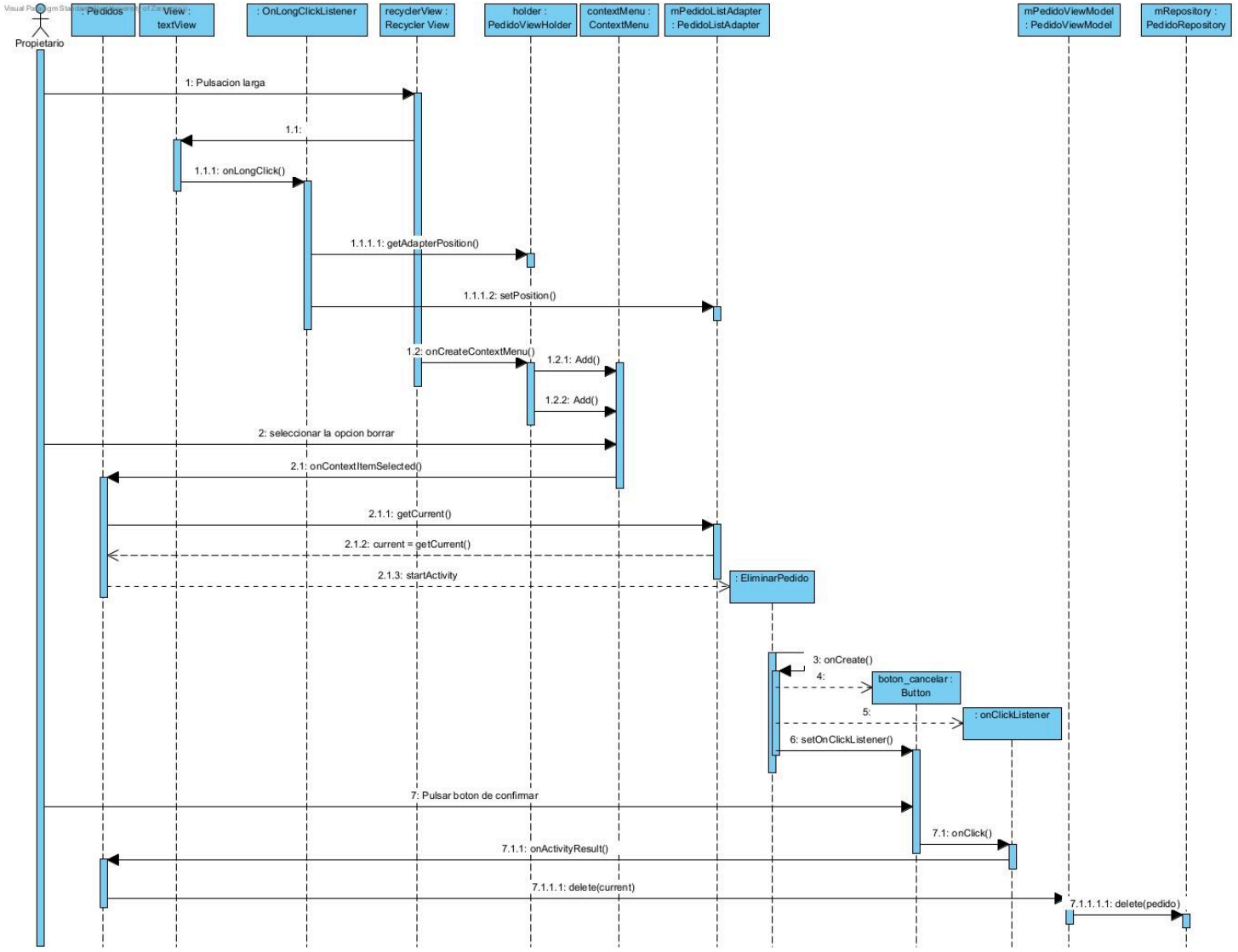


Editar pedido

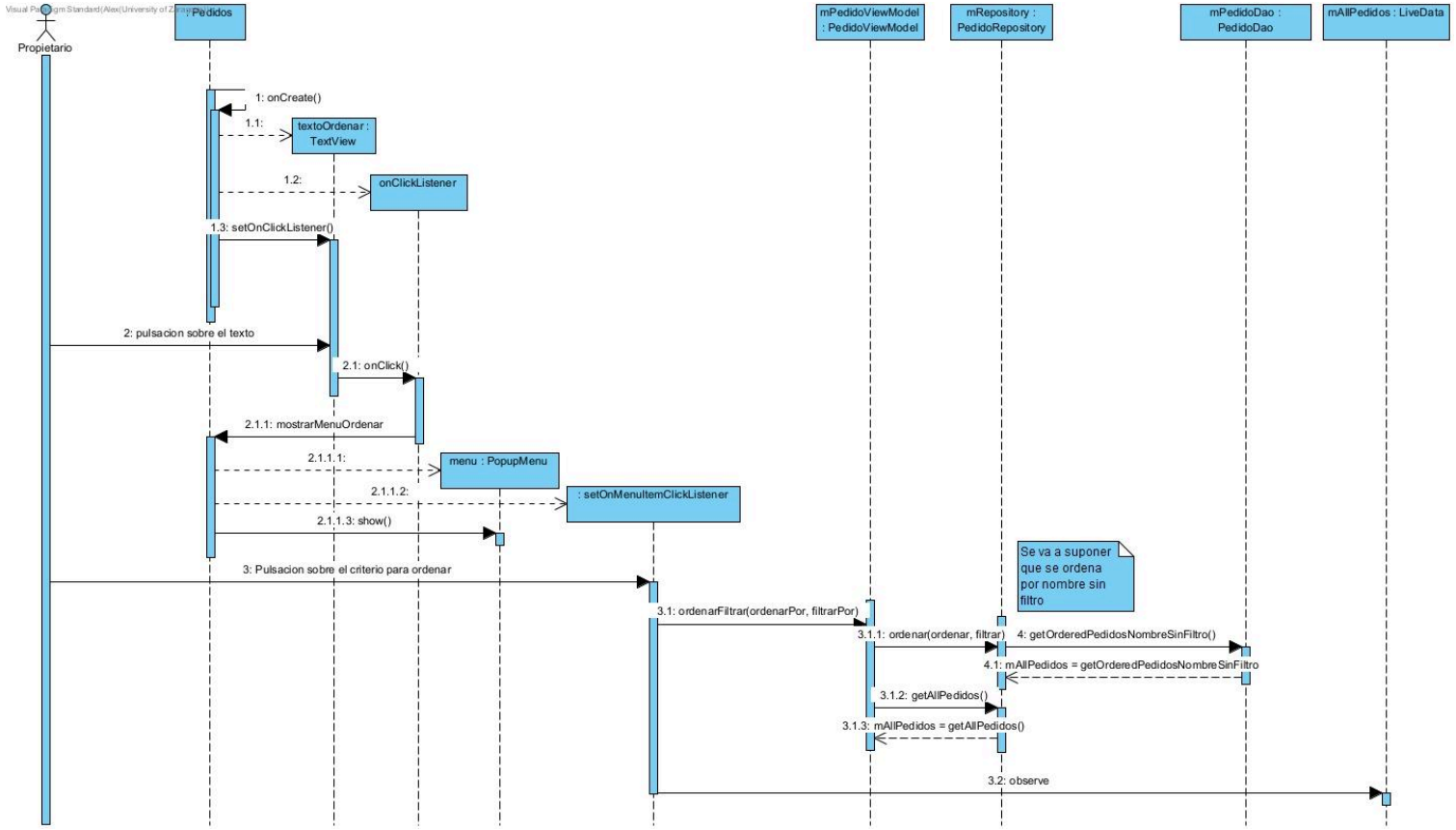


Para editar las raciones, se inicia la actividad Raciones el cual actualizará las raciones en tiempo real y lo insertará en base de datos

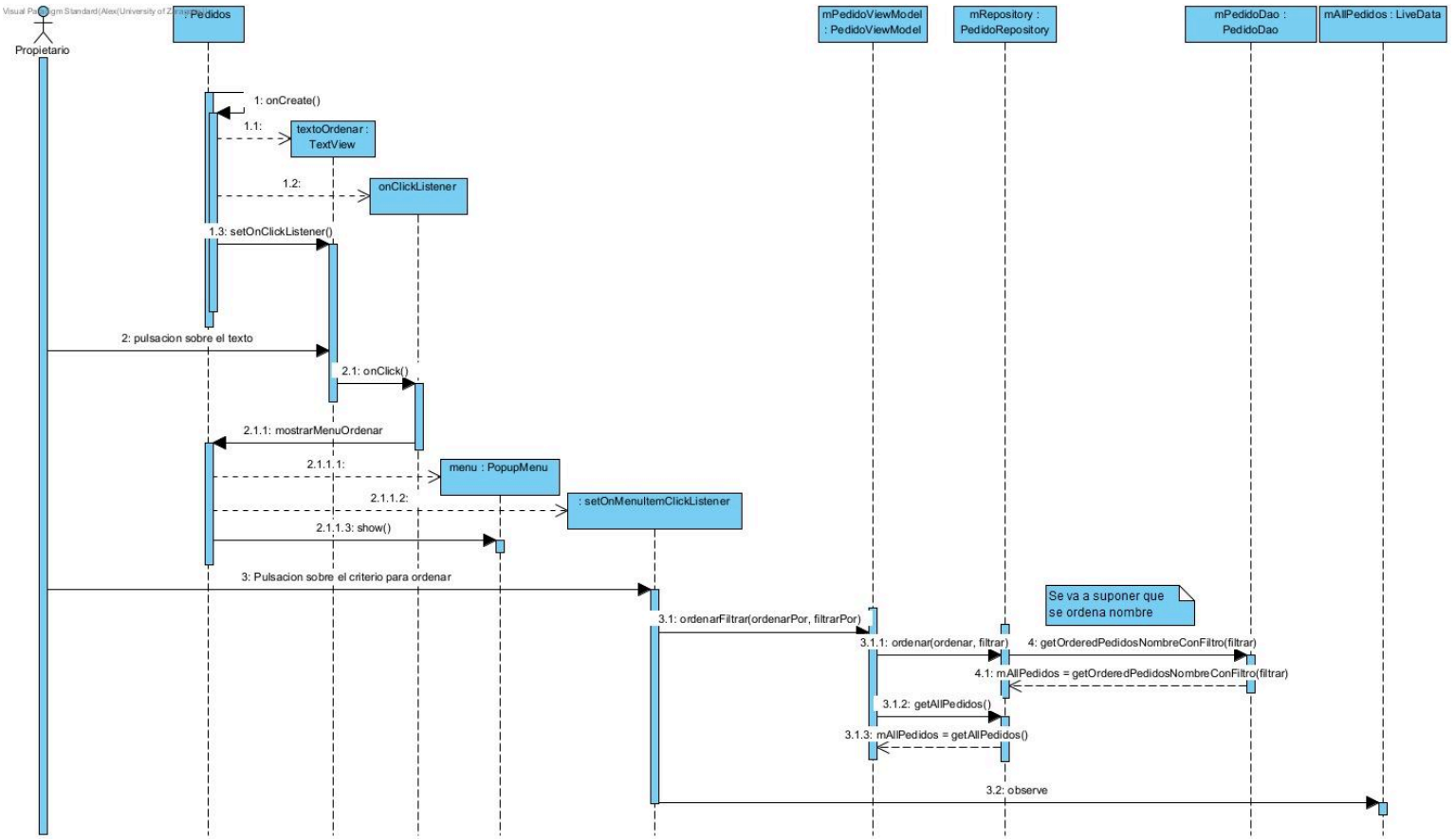
Borrar pedido



Ordenar pedidos



Filtrar pedidos



Notificar cliente por SMS

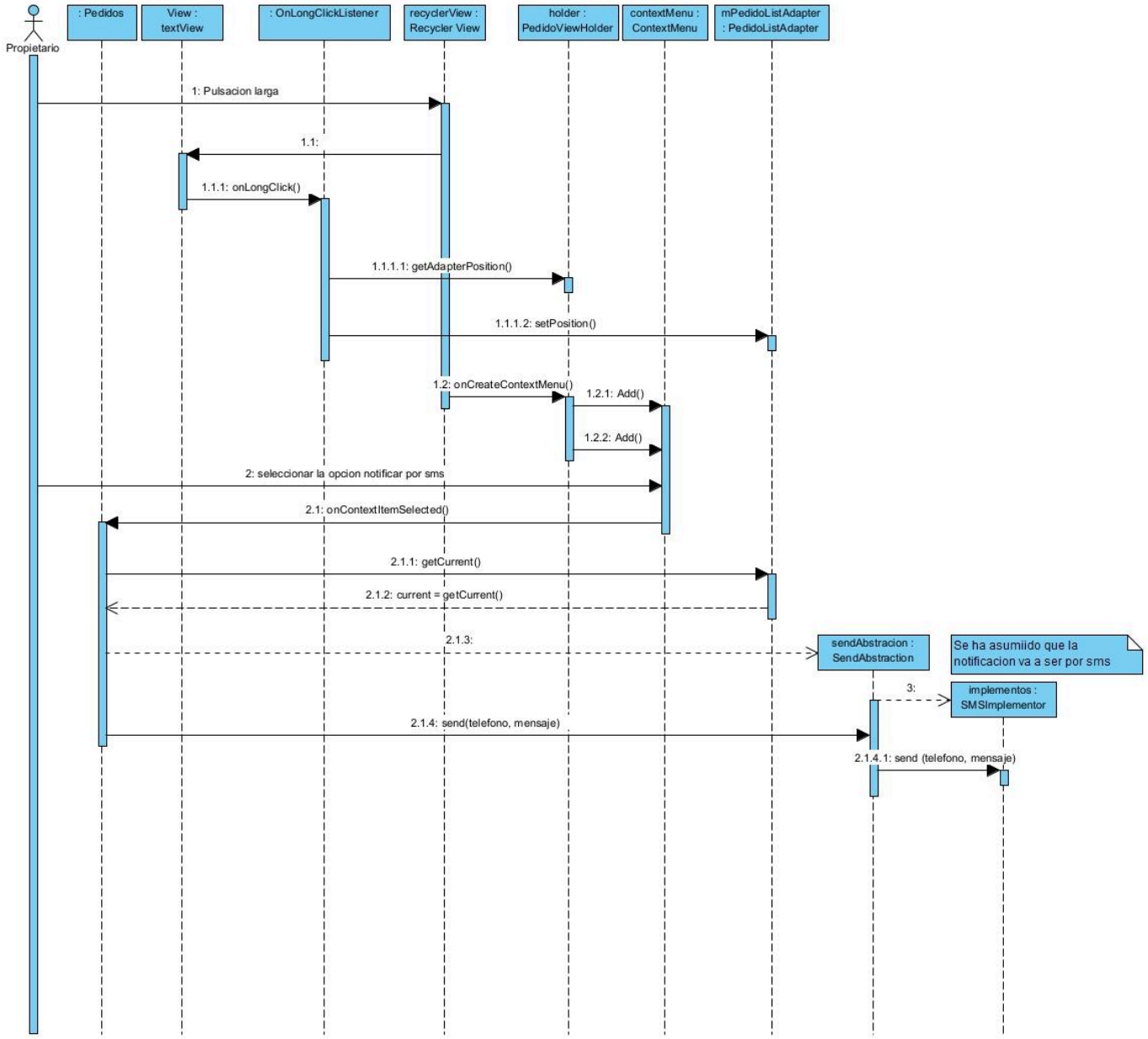
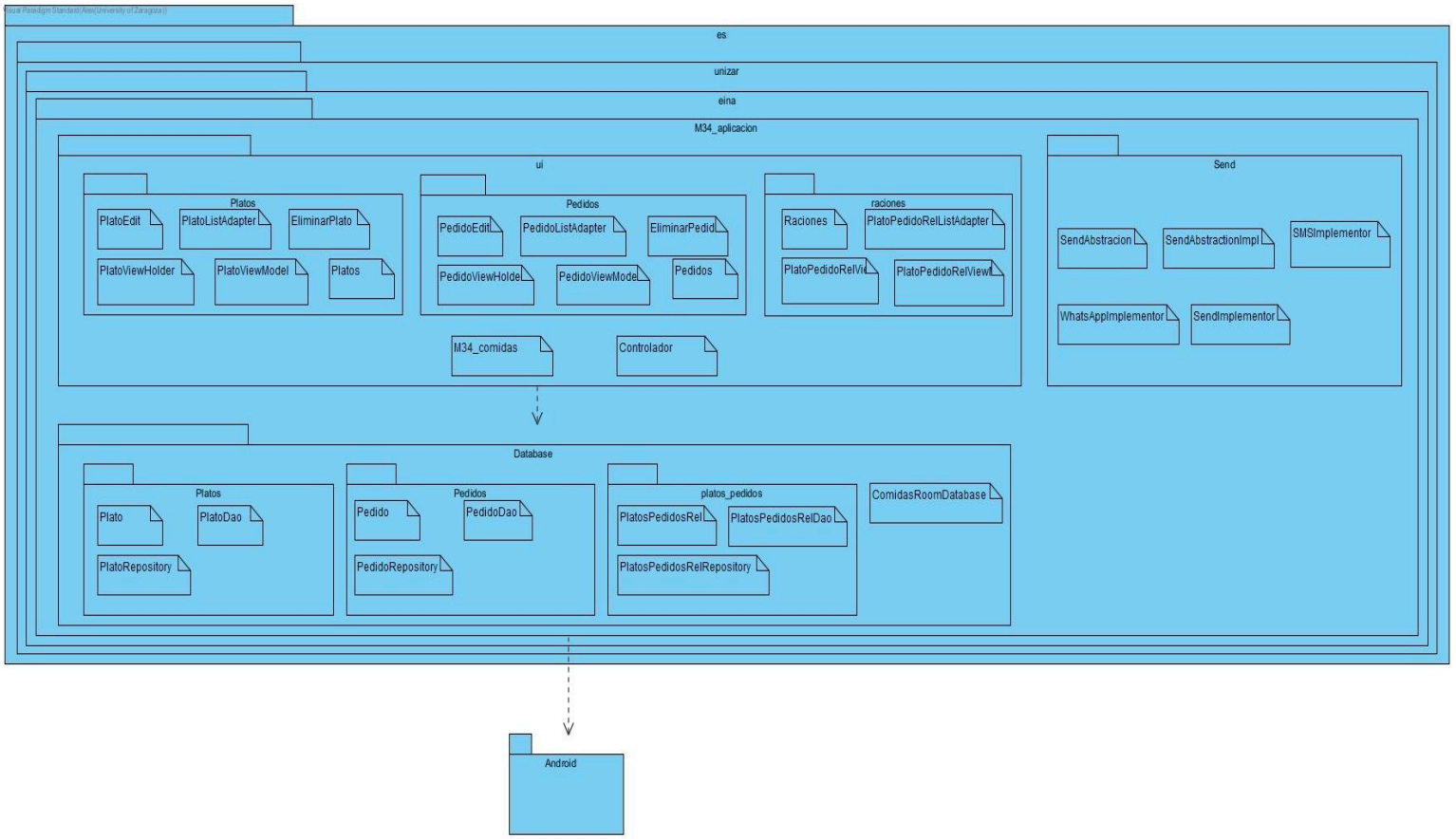


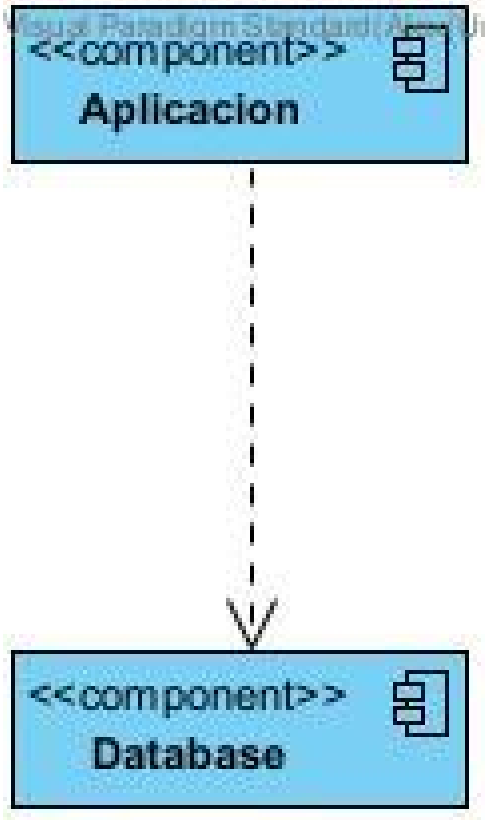
Diagrama de paquetes



Para organizar el código de la aplicación se han creado 2 paquetes o carpetas: ui y Database. En el paquete *ui* se encuentra la implementación de la lógica de la aplicación y las interfaces de usuario. Por otro lado, en el paquete *Database*, se encuentra todo lo relacionado con los datos así como las clases DAO, que hacen de intermediarias entre la base de datos y las interfaces. A su vez dentro de cada uno de estos 2 paquetes, hay una paquete con el código de los pedidos, otro con el de los platos y otro relacionado con las raciones.

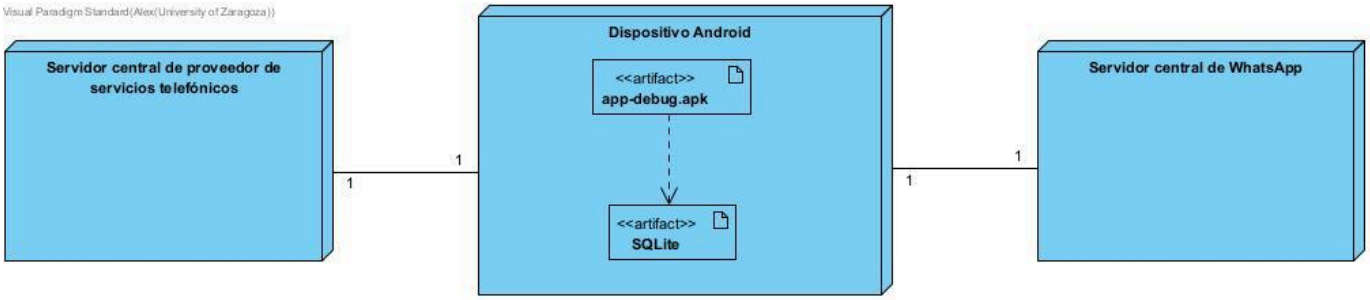
Para implementar la notificación al cliente se ha creado un subsistema o paquete llamado "es.unizar.eina.send" dentro. Este paquete contiene todas las interfaces y clases correspondientes al patrón Bridge para implementar dicha funcionalidad

Diagrama de componentes



Inicialmente, teníamos únicamente 2 componentes: la aplicación y la base de datos. En el componente Aplicación es donde se implementa todo lo relacionado a las interfaces de usuario y a la lógica de la aplicación. Por otro lado, en la base de datos se guardarán todos los pedidos y platos que se hayan creado.

Diagrama de despliegue



En el mismo dispositivo o máquina contiene la aplicación así como la base de datos *SQL Lite* para almacenar todos los datos

Para poder implementar la notificación al cliente se ha tenido que modificar el diagrama de despliegue añadiendo 2 nodos externos: Servidor central de WhatsApp (notificaciones por whatsApp) y el servidor central de proveedor de servicio telefónicos (notificaciones por sms)

PRUEBAS

Este apartado va a englobar aquellas pruebas y test que se hayan diseñado, construido y ejecutado para diversos casos donde podrían surgir errores en la aplicación.

Para dicho objetivo, hemos creado una nueva clase para ejecutar todos los métodos del caso de prueba y para evitar interrumpir la ejecución se ha utilizado la función de captura de excepciones para redirigirlas a un log.

Cabe destacar que tras la ejecución de los test, no se borra ningún plato, ni pedido, ni racion. Por lo tanto en la base de datos se encontrará todo lo creado durante los test

Pruebas unitarias de cajas negras

Pruebas para crear plato (método insert)

Definición de las clases de equivalencia para crear plato

Parametros	Clase de equivalencia valida	Clase de equivalencia no valida
plato.nombre	1) plato.nombre != null 2) plato.nombre.length() > 0	3) plato.nombre == null 4) plato.nombre.lenght() == 0
plato.descripcion	5) plato.descripcion != null 6) plato.descripcion.length() > 0	7) plato.descripcion == null 8) plato.descripcion.length() == 0
plato.precio	9) plato.precio != null	10) plato.precio == null
plato.categoria	11) plato.caterogia != null 12) plato.categoria.lenght() > 0	13) plato.categoria == null 14) plato.categoria.lenght() == 0

No se ha tenido en cuenta el id ya que este es autoincremental, es decir se genera automáticamente.

No se tiene en cuenta las diferentes casos de categoría. La aplicación utiliza un "RadioButtons" para elegir la categoría y siempre se inicializa en "primero".

Pruebas para crear plato

Entradas				Resultado	Clases cubiertas
nombre	Descripción	Precio	Categoría		
Casos de pruebas para clases de equivalencia válidas					
“pulpo”	“Ricos”	1.34	“primero”	1	1,2,5,6,9,11,12
Casos de pruebas para clases de equivalencia no válidas					
null	“Ricos”	1.34	“primero”	-1	3
“”	“Ricos”	1.34	“primero”	-1	4
“pulpo”	null	1.34	“primero”	-1	7
“pulpo”	“”	1.34	“primero”	-1	8
“pulpo”	“Ricos”	null	“primero”	-1	10
“pulpo”	“Ricos”	1.34	null	-1	13
“pulpo”	“Ricos”	1.34	“”	-1	14

Si el resultado es 1 o mayor que uno significa que el plato se ha creado correctamente. Si es -1 significa que no se ha creado el plato

Si se introdujera un plato a través de la aplicación (siguiendo los pasos) nunca se podría crear un nuevo plato si falta algún campo por rellenar. Si esto sucede salta un “Toast” e indicará que es necesario rellenar todos los formularios.

Para seleccionar la categoría, en la aplicación se realiza mediante “RadioButtons”. Nunca se podrá dar el caso de que no sea ni “primero”, ni “segundo” ni “tercero” ya que por defecto la categoría es “primero”. Si se introduce, de forma intencional, un valor que no será ninguno de los 3 comentados anteriormente, por defecto la aplicación lo tratara como “primero”

Editar o modificar plato (método update)

Definición de las clases de equivalencia para modificar plato

Parametros	Clase de equivalencia valida	Clase de equivalencia no valida
plato.id	1) plato.id => 0 2) plato.id != null	3) plato.id < 0
plato.nombre	4) plato.nombre != null 5) plato.nombre.length() > 0	6) plato.nombre == null 7) plato.nombre.lenght() == 0
plato.descripcion	8) plato.descripcion != null 9) plato.descripcion.length() > 0	10) plato.descripcion == null 11) plato.descripcion.length() == 0
plato.precio	12) plato.precio != null	13) plato.precio == null
plato.categoria	15) plato.caterogia != null 16) plato.categoria.lenght() > 0	17) plato.categoria == null 18) plato.categoria.lenght() == 0

El id nunca puede ser null.
No se tiene en cuenta las diferentes casos de categoría. La aplicación utiliza un "RadioButtons" para elegir la categoría y siempre se inicializa en "primero".

Pruebas para modificar plato

Entradas					Resultado	Clases cubiertas
Id	nombre	Descripción	Precio	Categoría		
Casos de pruebas para clases de equivalencia válidas						
1	"pulpo"	"Ricos"	1.34	"primero"	1	1,2,5,6,9,10,13,15,16
Casos de pruebas para clases de equivalencia no válidas						
1	null	"Ricos"	1.34	"primero"	0	6
1	""	"Ricos"	1.34	"primero"	0	7
1	"pulpo"	null	1.34	"primero"	0	10
1	"pulpo"	""	1.34	"primero"	0	11
1	"pulpo"	"Ricos"	null	"primero"	0	13
1	"pulpo"	"Ricos"	1.34	null	0	17
1	"pulpo"	"Ricos"	1.34	""	0	18

-4	“pulpo”	“Ricos”	1.34	“primero”	0	3

Si el resultado es mayor es 0 (false) significa que no se ha actualizado nada. Por otro lado si es 1(true) significa que si se ha modificado

Si se introdujera un plato a través de la aplicación (siguiendo los pasos) nunca se podría crear un nuevo plato si falta algún campo por rellenar. Si esto sucede salta un “Toast” e indicará que es necesario rellenar todos los formularios.

Para seleccionar la categoría, en la aplicación se realiza mediante “RadioButtons”. Nunca se podrá dar el caso de que no sea ni “primero”, ni “segundo” ni “tercero” ya que por defecto la categoría es “primero”. Si se introduce, de forma intencional, un valor que no será ninguno de los 3 comentados anteriormente, por defecto la aplicación lo tratara como “primero”

Eliminación de platos (método delete)

Definición de las clases de equivalencia para modificar plato

Parámetros	Clase de equivalencia valida	Clase de equivalencia no valida
plato.id	1) plato.id >= 0	2) plato.id < 0

Para borrar un plato lo único importante es el id

Pruebas Entradasde eliminar plato

					Resultado	Clases cubiertas
Id	nombre	Descripción	Precio	Categoría		
Casos de pruebas para clases de equivalencia válidas						
1	“pulpo”	“Ricos”	1.34	“primero”	1	1
Casos de pruebas para clases de equivalencia no válidas						
-4	“pulpo”	“Ricos”	1.34	“primero”	-1	2

Si el resultado es 1 significa que se ha borrado. Si es 0 no se ha borrado nada

Pruebas para crear pedido (método insert)

Definición de las clases de equivalencia para crear plato

Parametros	Clase de equivalencia valida	Clase de equivalencia no valida
pedido.nombre	1) Pedido.nombre != null 2) plato.nombre.length() > 0	3) Pedido.nombre == null 4) Pedido.nombre == 0
pedido.telefono	5) pedido.telefono != null 6) pedido.telefono.legnth() > 0	7) pedido.telefono == null 8) pedido.telefono.length() == 0
Pedido.fecha	9) Pedido.fecha != null 10) pedido.fecha.length() > 0	11) Pedido.fecha == null 12) Pedido.fecha.lenght == 0
Pedido.hora	13) Pedido.hora != null 14) Pedido.hora.length() > 0	15) Pedido.hora == null 16) Pedido.hora.lenght() == 0
Pedido.estado	17) Pedido.estado != null 18) Pedido.estado.lenght() > 0	19) Pedido.estado == null 20) Pedido.estado.lenght() == 0

No se ha tenido en cuenta el id ya que este es autoincremental, es decir se genera automáticamente. El precioTotal del pedido aqui es irrelevante ya que para crear el pedido no necesitas el preciototal, ya que este sera rellenado al meter las raciones.

No considero el formato de la fecha ni de la hora ya que la aplicación utiliza un “DataPicker” de android para introducir ambos casos. Por eso siempre se va a respetar el mismo formato. Tampoco consideró las diferentes opciones del estado ya que para introducirlo en la aplicación se utiliza “RadioButtons” y por defecto siempre se inicializa en primero.

Pruebas

Entradas					Resultado	Clases cubiertas
nombre	telefono	fecha	hora	estado		
Casos de pruebas para clases de equivalencia válidas						
“Pedro”	“6458293”	“13/01/2024”	“20:30”	“solicitado”	1	1,2,5,6,9,10,13,14,17,18
Casos de pruebas para clases de equivalencia no válidas						
null	“6458293”	“13/01/2024”	“20:30”	“solicitado”	-1	3
“”	“6458293”	“13/01/2024”	“20:30”	“solicitado”	-1	4
“Pedro”	null	“13/01/2024”	“20:30”	“solicitado”	-1	7

“Pedro”	“”	“13/01/2024”	“20:30”	“solicitado”	-1	8
“Pedro”	“6458293”	null	“20:30”	“solicitado”	-1	11
“Pedro”	“6458293”	“”	“20:30”	“solicitado”	-1	12
“Pedro”	“6458293”	“13/01/2024”	null	“solicitado”	-1	15
“Pedro”	“6458293”	“13/01/2024”	“”	“solicitado”	-1	16
“Pedro”	“6458293”	“13/01/2024”	“20:30”	null	-1	19
“Pedro”	“6458293”	“13/01/2024”	“20:30”	“”	-1	20

Si el resultado es 1 o mayor que uno significa que el pedido se ha creado correctamente. Si es -1 significa que no se ha creado el pedido

Si se introdujera un pedido a través de la aplicación (siguiendo los pasos) nunca se podría crear un nuevo pedido si falta algún campo por rellenar. Si esto sucede salta un “Toast” e indicará que es necesario rellenar todos los formularios.

Para seleccionar el estado, en la aplicación se realiza mediante “RadioButtons”. Nunca se podrá dar el caso de que no sea ni “solicitado”, ni “preparado” ni “recogido” ya que por defecto la categoría es “solicitado”. Si se introduce, de forma intencional, un valor que no será ninguno de los 3 comentados anteriormente, por defecto la aplicación lo tratara como “solicitado”.

Modificar pedido (metodo update)

Definición de las clases de equivalencia para modificar

Parametros	Clase de equivalencia valida	Clase de equivalencia no valida
pedido.id	1) pedido.id => 0	2) pedido.id < 0
pedido.nombre	3) Pedido.nombre != null 4) plato.nombre.length() > 0	5) Pedido.nombre == null 6) Pedido.nombre == 0
pedido.telefono	7) pedido.telefono != null 8) pedido.telefono.legnth() > 0	9) pedido.telefono == null 10) pedido.telefono.length() == 0
Pedido.fecha	11) Pedido.fecha != null 12) pedido.fecha.length() > 0	13) Pedido.fecha == null 14) Pedido.fecha.lenght == 0
Pedido.hora	15) Pedido.hora != null 16) Pedido.hora.length() > 0	17) Pedido.hora == null 18) Pedido.hora.lenght() == 0
Pedido.estado	19) Pedido.estado != null 20) Pedido.estado.lenght() > 0	21) Pedido.estado == null 22) Pedido.estado.lenght() == 0

El precioTotal del pedido aquí es irrelevante ya que para editar el pedido no necesitas saber el , ya que este será actualizado si se actualizan los pedidos.

No considero el formato de la fecha ni de la hora ya que la aplicación utiliza un “DataPicker” de android para introducir ambos casos. Por eso siempre se va a respetar el mismo formato. Tampoco consideró las diferentes opciones del estado ya que para introducirlo en la aplicación se utiliza “RadioButtons” y por defecto siempre se inicializa en primero.

Pruebas

Entradas						Resultado	Clases cubiertas
id	nombre	telefono	fecha	hora	estado		
Casos de pruebas para clases de equivalencia válidas							
3	“Pedro”	“6458293”	“13/01/2024”	“20:30”	“solicitado”	1	1,3,4,7,8,11,12,15,16 ,19,20
Casos de pruebas para clases de equivalencia no válidas							
3	null	“6458293”	“13/01/2024”	“20:30”	“solicitado”	0	5
3	“”	“6458293”	“13/01/2024”	“20:30”	“solicitado”	0	6

3	“Pedro”	null	“13/01/2024”	“20:30”	“solicitado”	0	9
3	“Pedro”	“”	“13/01/2024”	“20:30”	“solicitado”	0	10
3	“Pedro”	“6458293”	null	“20:30”	“solicitado”	0	13
3	“Pedro”	“6458293”	“”	“20:30”	“solicitado”	0	14
3	“Pedro”	“6458293”	“13/01/2024”	null	“solicitado”	0	17
3	“Pedro”	“6458293”	“13/01/2024”	“”	“solicitado”	0	18
3	“Pedro”	“6458293”	“13/01/2024”	“20:30”	null	0	21
3	“Pedro”	“6458293”	“13/01/2024”	“20:30”	“”	0	22
-4	“Pedro”	“6458293”	“13/01/2024”	“20:30”	“solicitado”	0	2

Si el resultado es mayor es 0 (false) significa que no se ha actualizado nada. Por otro lado si es 1(true) significa que si se ha modificado

Si se introdujera un pedido a través de la aplicación (siguiendo los pasos) nunca se podría crear un nuevo pedido si falta algún campo por rellenar. Si esto sucede salta un “Toast” e indicará que es necesario rellenar todos los formularios.

Para seleccionar el estado, en la aplicación se realiza mediante “RadioButtons”. Nunca se podrá dar el caso de que no sea ni “solicitado”, ni “preparado” ni “recogido” ya que por defecto la categoría es “solicitado”. Si se introduce, de forma intencional, un valor que no será ninguno de los 3 comentados anteriormente, por defecto la aplicación lo tratara como “solicitado”.

Eliminar pedido (método delete)

Definición de las clases de equivalencia para eliminar plato

Parámetros	Clase de equivalencia válida	Clase de equivalencia no válida
pedido.id	1) Pedido.id >= 0	2) Pedido.id < 0

Para eliminar solo es necesario el id

Pruebas de eliminar pedido

Entradas						Resultado	Clases cubiertas
Id	nombre	Telefono	Fecha	Hora	Estado		
Casos de pruebas para clases de equivalencia válidas							
1	“Pedro”	“6458293”	“13/01/2024”	“20:30”	“solicitado”	1	1
Casos de pruebas para clases de equivalencia no válidas							
-4	“Pedro”	“6458293”	“13/01/2024”	“20:30”	“solicitado”	0	2

Si el resultado es 1 significa que se ha borrado. Si es 0 no se ha borrado nada

Añadir ración y editar ración (método insert y update)

Clases de equivalencia de insertar y actualizar racion

Parametros	Clases de equivalencia validas	Clases de equivalencia no validas
racion.cantidad	1) racion.cantidad >= 0	2) racion.cantidad < 0
racion.precioCrear	3) racion.precioCrear != null 4) racion.precioCrear >= 0	5) racion.precioCrear == null 6) racion.precioCrear < 0

Tanto platoId como pedidoS son claves foráneas de plato y pedido respectivamente. Por esto mismo no se puede realizar pruebas sobre ellos, ya que si se introduce un id que no existe se va a producir una excepción de integridad de la base de datos. Si se pruebas claves negativas(cosa que no es posible) saltara una excepcion ya que no existe ninguna clave. Para hacer las pruebas se asume que existe el id.

La cantidad no puede ser null ya que es de tipo int.

Pruebas insertar y actualizar

Entradas				Resultado	Clases cubiertas
platoId	pedidoId	cantidad	precioCrear		
Casos de pruebas para clases de equivalencia válidas					
1	1	4	13.32	1	1,3,4
Casos de pruebas para clases de equivalencia no válidas					
1	1	-4	13.32	-1	2
1	1	4	null	-1	5
1	1	4	-4.0	-1	6

Esta tabla corresponde con la tabla de pruebas de insertar. La tabla de modificar es exactamente igual pero cambiando el -1 por un 0 (no se ha modificado ninguna fila).

Eliminar Ración

Para eliminar la ración basta con que los id coincidan con los de plato y pedido respectivamente. Al ser restricciones de la base de datos, si no se encuentra la clave salta una excepción de integridad de la base de datos

Entradas				Resultado	Clases cubiertas
platoId	pedidoId	cantidad	precioCrear		
Casos de pruebas para clases de equivalencia válidas					
1	1	4	13.32	1	-

Pruebas sobre carga

Se ha creado otro botón con texto “ejecutar sobrecarga” para poder ejecutar este tipo de test de forma independiente a los test de caja negra.

Durante la ejecución de estas pruebas, se observó que el máximo de caracteres ha sido de 10000000 momento en el cual se alcanzaba la capacidad máxima. Se obtiene un error de “OutOfMemory” y se cierra la aplicación