INGENIERÍA DEL SOFTWARE

Desarrollo de una aplicación para la gestión de pedidos en una tienda de preparación de comidas

Grupo M35

Hecho por

ÁLVARO DE FRANCISCO NIEVAS, 838819@unizar.es
GUILLERMO BAJO LABORDA, 842748@unizar.es

Universidad Zaragoza
1542

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

Índice

Resumen

A lo largo de esta primera práctica se ha realizado el análisis, diseño, implementación y pruebas de una aplicación móvil que permite que el propietario de una tienda de preparación de comidas pueda gestionar los pedidos que le solicitan sus clientes por teléfono. El trabajo se ha realizado mayoritariamente con ambos componentes del grupo presentes para poder debatir diversos temas y tomar decisiones consensuadas y más razonadas bajo distintos puntos de vista.

Introducción y objetivos

Para realizar dicho trabajo, se ha utilizado la herramienta CASE Visual Paradigm. Primeramente ha realizado la captura de los requisitos de la aplicación, así como su representación mediante un catálogo de requisitos y la definición de casos de uso asociados a dichos requisitos. También se ha realizado un análisis de la aplicación mediante el diseño del modelado estático, mediante diagramas de clase, y del modelado dinámico, mediante diagramas de secuencia.

Requisitos

Catálogo de requisitos

En primer lugar concretamos los requisitos de nuestra aplicación. Estos se dividen en requisitos funcionales, es decir , una función del sistema de software o sus componentes; y no funcionales, que especifica criterios que pueden usarse para juzgar la operación de un sistema en lugar de sus comportamientos específicos. Además añadimos restricciones sobre nuestro diseño. Sobre esta base construiremos el diagrama de casos de uso y a su vez los diagramas de clase y de secuencias.

CATÁLOGO DE REQUISITOS	
Código	Descripción
RF-1	La aplicación debe permitir la creación de platos
RF-2	Los platos deben tener nombre, descripción en texto libre de ingredientes, categoría (primero, segundo y postre) y precio en euros.
RF-3	La aplicación debe permitir la consulta de un listado de platos creados previamente.
RF-4	La aplicación debe permitir la ordenación del listado de platos por categoría, nombre o ambos criterios.
RF-5	La aplicación debe permitir la modificación de los platos previamente creados.
RF-6	La aplicación debe permitir la eliminación de los platos previamente creados.
RF-7	La aplicación debe permitir la creación de un pedido.
RF-8	Los pedidos deben tener un nombre de cliente, su número de teléfono móvil, la fecha y hora a la que se recogerá el pedido y una selección del número de raciones que se necesitan de algunos de los platos previamente creados
RF-9	La fecha y hora deben ser posteriores a la fecha y hora actual, y deben estar comprendidas entre el horario de recogida de pedidos de la tienda (de martes a domingo entre las 19:30 y las 23:00).
RF-10	El pedido debe tener un estado inicial de SOLICITADO por defecto, pero puede cambiar a PREPARADO o RECOGIDO.
RF-11	La aplicación debe permitir la consulta de un listado de pedidos previamente creados.

RF-12	La aplicación debe permitir la ordenación del listado de pedidos por nombre, número de móvil o fecha y hora de recogida
RF-13	La aplicación debe permitir el filtrado de la lista de pedidos según el estado de estos.
RF-14	La aplicación debe permitir modificar pedidos previamente creados.
RF-15	La aplicación debe permitir el cálculo del precio total del pedido automático en la creación del mismo, o en su modificación si se realiza una selección distinta de platos o número de raciones.
RF-16	El precio total se debe mantener aunque se modifique el precio por ración después de la última actualización de platos y raciones en el pedido
RF-17	El envío al móvil del cliente de la información del pedido, incluido el precio
RF-18	La aplicación debe permitir la eliminación de los pedidos previamente creados.
RNF-1	La aplicación debe ser capaz de manejar al menos 100 platos y 2000 pedidos almacenados

CATÁLOGO DE RESTRICCIONES	
Número	Descripción
1	La aplicación debe ser compatible para dispositivos que utilicen el sistema operativo Android

Diagrama de casos de uso

Tras realizar el catálogo de requisitos, diseñamos el diagrama de casos de uso que representa los requisitos funcionales de la aplicación.. El diagrama de la aplicación en cuestión es el siguiente:

Descripción textual de los casos de uso

Una vez descrita la relación entre los distintos usos de nuestro sistema, se procede a la descripción textual de los flujos principales y alternativos de los casos.

Creación de un nuevo plato

- Actor: Propietario
- Flujo de eventos principal:
  a. El administrador abre la aplicación.
  b. El sistema muestra la opción de crear un nuevo plato, junto a otras. Seleccionamos "crear nuevo plato".
  c. El sistema muestra una lista de campos a rellenar. Estos son: nombre, descripción categoría (a elegir entre primero, segundo y postre) y precio.
  d. Una vez completados los campos, el administrador confirma la creación del plato.
  e. El sistema nos devuelve a la pantalla principal.

Modificación de un Plato

- Actor: Propietario
- Flujo de eventos principal:
  a. El administrador abre la aplicación.
  b. El sistema muestra una lista de platos existentes para que el administrador seleccione el plato que desea modificar.
  c. El administrador elige un plato de la lista.
  d. El sistema muestra los campos del plato seleccionado (nombre, descripción, lista de ingredientes, categoría y precio) para que el administrador los modifique según sea necesario.
  e. Una vez realizadas las modificaciones, el administrador confirma la actualización del plato.
  f. El sistema devuelve al administrador a la pantalla principal.

Eliminación de un Plato

- Actor: Propietario
- Flujo de eventos principal:
  a. El administrador abre la aplicación.
  b. El sistema muestra una lista de platos existentes (una lista vacía en caso de que no existan platos) para que el administrador seleccione el plato que desea eliminar.
  c. El administrador elige un plato de la lista.
  d. El sistema muestra una confirmación de eliminación y solicita al administrador que confirme la acción.
  e. El administrador confirma la eliminación del plato.
  f. El sistema devuelve al administrador a la pantalla principal.

Listar Platos Existentes

- Actor: Propietario
- Flujo de eventos principal:
  a. El administrador abre la aplicación.
  b. El sistema muestra el listado de platos existentes.
  c. El administrador puede seleccionar el criterio de ordenación.
- Flujo de eventos alternativo: se selecciona ordenación
  a. El administrador selecciona el criterio de ordenación.
  b. El sistema genera una lista de todos los platos existentes en la base de datos bajo el orden especificado.

Crear Pedido

- Actor: Propietario
- Flujo de eventos principal:
  a. El administrador abre la aplicación.
  b. El sistema muestra la opción de "Crear Pedido" en la pantalla principal. El administrador selecciona esta opción.
  c. El administrador ingresa el nombre, número móvil del cliente, fecha y hora de recogida del pedido.
  d. El sistema valida la fecha y hora del pedido.
  e. El administrador selecciona los platos previamente creados y especifica el número de raciones de cada uno.
  f. El sistema calcula automáticamente el precio total del pedido.
  g. El sistema asigna el estado "SOLICITADO" al pedido.
  h. El sistema envía la información del pedido (incluido el precio) al número móvil del cliente.
- Flujo de eventos alternativo: pedido fuera de hora

a. En el paso d) del flujo principal, la hora y fecha de recogida no se encuentra en el horario permitido. El pedido no se crea.

Modificar Pedido

- Actor: Propietario
- Flujo de eventos principal:
  a. El administrador abre la aplicación.
  b. El sistema muestra la opción de "Modificar Pedido" en la pantalla principal. El cliente selecciona esta opción.
  c. El administrador busca y selecciona el pedido que desea modificar de una lista de pedidos previamente creados.
  d. El sistema muestra los detalles del pedido, incluyendo nombre del cliente, número móvil, fecha y hora de recogida, platos seleccionados y estado.
  e. El administrador puede realizar modificaciones, como cambiar la fecha y hora de recogida, la selección de platos o el número de raciones.
  f. El sistema recalcula automáticamente el precio total del pedido.

g. El administrador confirma las modificaciones.
h. El sistema actualiza el estado del pedido según sea necesario (puede cambiar a "PREPARADO" o "RECOGIDO").
i. El sistema envía la información actualizada del pedido (incluido el precio) al número móvil del cliente.
j. El sistema devuelve al cliente a la pantalla principal.

● Flujo de eventos alternativo: pedido fuera de hora

a. En el paso c) del flujo principal, la hora y fecha de recogida no se encuentra en el horario permitido. El sistema no permite la modificación del pedido.

Eliminar pedido

● Actor: Propietario

● Flujo de eventos principal:

a. El administrador abre la aplicación.
b. El sistema muestra la opción de "Eliminar Pedido" en la pantalla principal. El cliente o administrador selecciona esta opción.
c. El administrador busca y selecciona el pedido que desea eliminar de una lista de pedidos previamente creados.
d. El administrador confirma la eliminación del pedido.
e. El sistema devuelve al cliente o administrador a la pantalla principal.

Listar pedidos

● Actor: Propietario

● Flujo de eventos principal:

a. El administrador abre la aplicación.
b. El sistema genera una lista de todos los pedidos existentes, permitiendo ordenarlos por nombre de cliente, número de móvil o fecha y hora de recogida, y filtrarlos para visualizar solo pedidos con un estado específico (SOLICITADO, PREPARADO o RECOGIDO).

● Flujo de eventos alternativo 1: se selecciona filtrado

a. En el paso (b, el administrador selecciona el criterio de filtrado.
b. El sistema genera una lista de todos los pedidos existentes bajo el filtro especificado.

● Flujo de eventos alternativo 2: se selecciona ordenación

a. En el paso (b, el administrador selecciona el criterio de ordenación.
b. El sistema genera una lista de todos los pedidos existentes bajo el orden especificado.

Análisis

Una vez completada la parte de captura de requisitos, se procede a la fase de análisis, en la cual se realizará tanto el análisis estático como el dinámico de la aplicación.

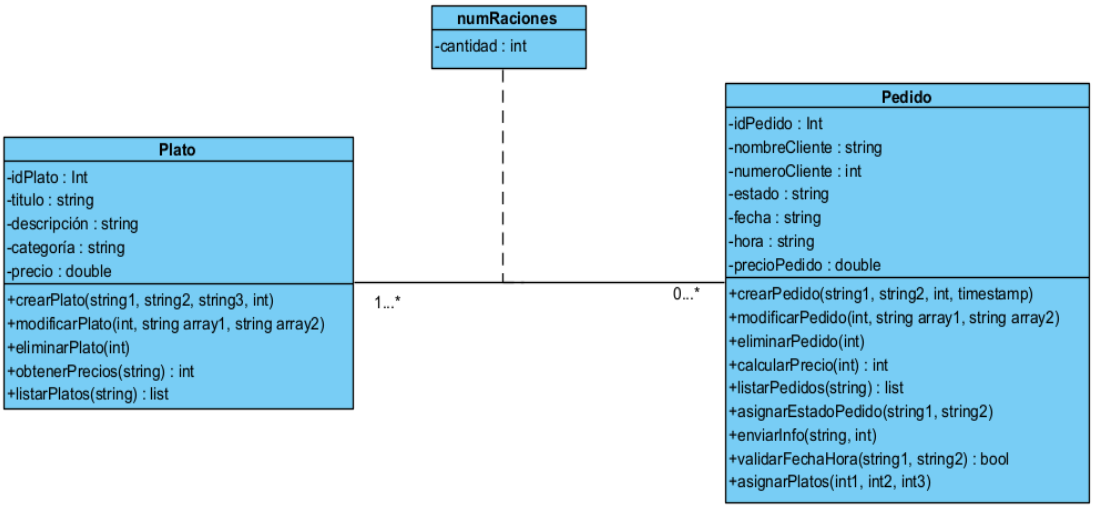
Modelo estático

La realización de un diagrama de clases es una forma muy útil de representar la información necesaria para gestionar tanto platos como pedidos. Se almacenan los diversos atributos y operaciones de estos, así como las relaciones que pueden existir entre diversas clases.

Se han diseñado tres clases: *Plato*, *Pedido* y *numRaciones*. Las clases Plato y Pedido sirven para representar la información básica de la aplicación, pero es necesaria la clase *numRaciones* para representar el listado de los platos que contiene un pedido, con las respectivas cantidades de cada uno de los platos. Si se optase por poner el atributo *listadoPlatos* de la clase Pedido como una lista, no se podría almacenar la cantidad de cada uno de los platos, por esto la necesidad de crear una nueva clase.

Respecto a las operaciones de cada una de las clases, si bien cierto es que en la aplicación implementada no se encuentran representadas de forma idéntica, hay implementadas diversos métodos que realizan la labor de estas operaciones. Estas han sido implementadas a partir de las operaciones diseñadas en el modelo estático que han servido como modelo conceptual a la hora de implementar las diferentes funcionalidades que realizan la tarea conceptual.

El diagrama de clases en cuestión es el siguiente



Las clases creadas son las siguientes:

- **Plato**: Representa los platos de la aplicación, se guardan su nombre, descripción, categoría y precio. La clase tiene varias operaciones, las principales son la creación, modificación y eliminación de un plato, la obtención del precio y el listado de platos existentes, entre otras.
- **Pedido**: Representa los pedidos de la aplicación, se guardan el nombre del cliente, su número de teléfono, el estado del pedido, el listado de los platos del pedido y la fecha y hora del mismo. Las operaciones disponibles sobre esta clase son la creación, modificación y eliminación de pedido, así como el cálculo del precio del pedido y el listado de los pedidos realizados,la validación de modificaciones y de fecha y hora, la asignación de estado al pedido, el envío de información al cliente y la asignación de platos.
- **numRaciones**: Utilizada para representar la cantidad de raciones de un plato que hay en un pedido determinado.

Modelo dinámico

Tras haber realizado el análisis estático de la aplicación, se procede a realizar el dinámico, el cual describe los flujos de eventos que aparecen en los casos de uso.

Diagrama de secuencia de crearPlato

En este primer diagrama se crea un plato con los parámetros correspondientes. Cabe destacar que para los diagramas se ha considerado que se encuentra la opción deseada ya seleccionada, no se han de poner en el diagrama acciones como que se ha seleccionado la opción de creación de platos o, por ejemplo, en la eliminación de un plato indicar que se ha seleccionado el plato a eliminar.

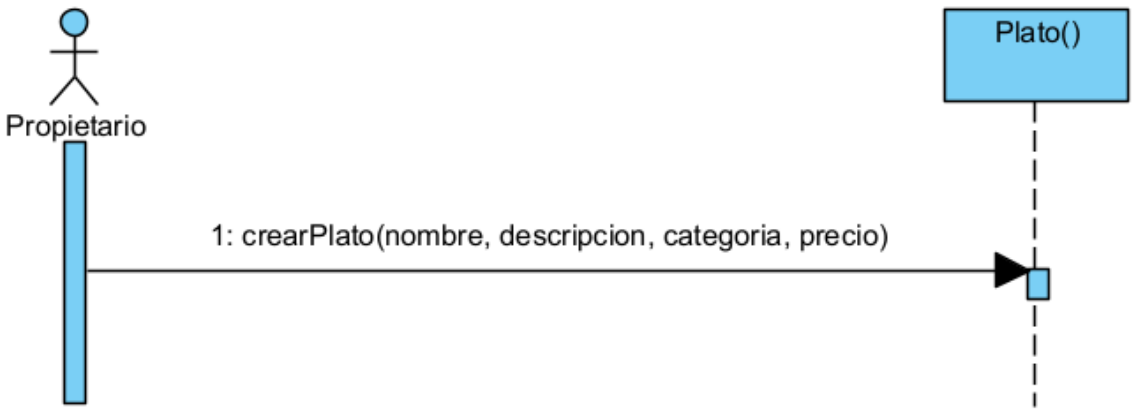


Diagrama de secuencia de modificarPlato

En este diagrama de secuencia, el administrador modifica un plato, pasando dos arrays a la clase plato, uno de propiedades a modificar y otro de modificaciones a realizar en cada una de las propiedades seleccionadas.

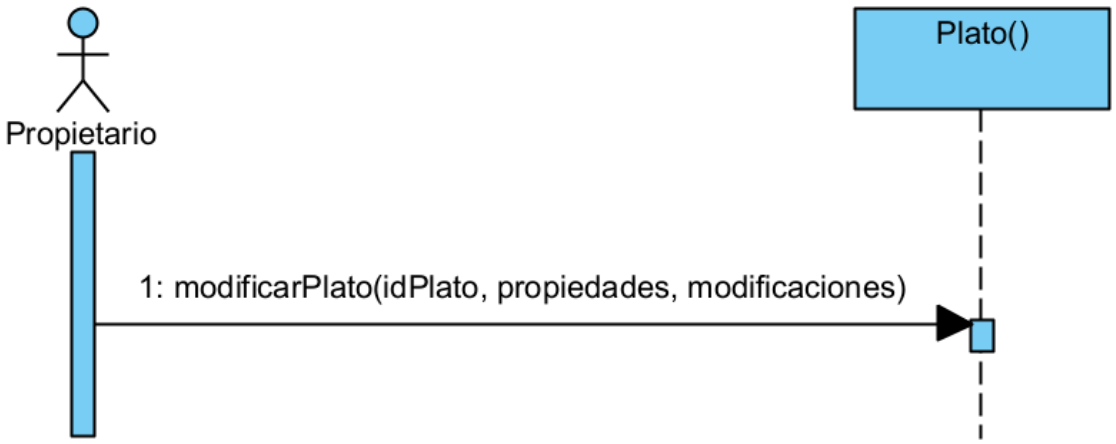


Diagrama de secuencia de eliminarPlato

En este sencillo diagrama simplemente el administrador utiliza la operación de eliminar un plato en concreto. Tras la llamada a la operación, la línea de vida del plato muere.

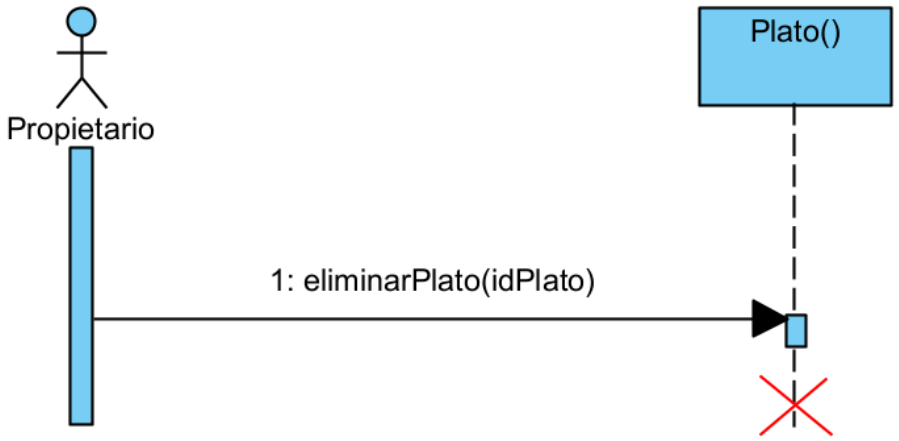


Diagrama de secuencia de listarPlatos

En este diagrama el administrador solicita el listado de platos y estos son mostrados por pantalla. Si el criterio es nulo se ordena de forma predeterminada, si es menor de menor a mayor.

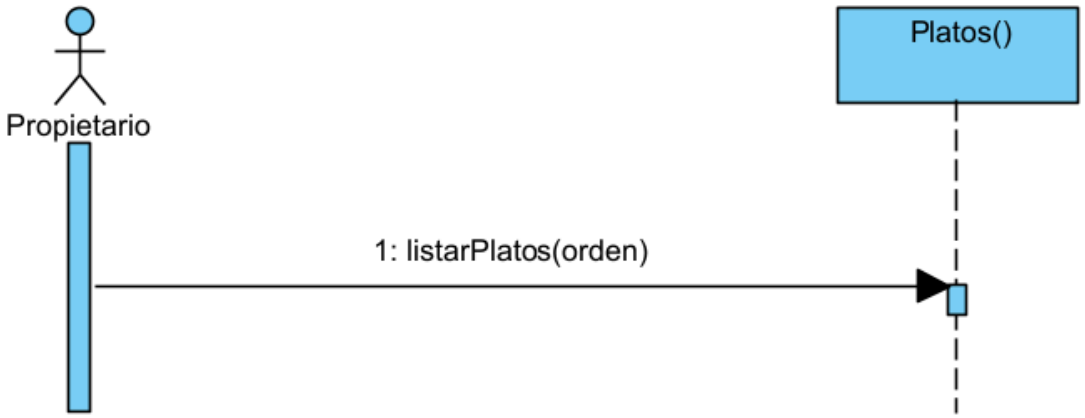


Diagrama de secuencia de crearPedido

En este diagrama el administrador procede a crear el pedido introduciendo los datos correspondientes (nombre del cliente, número de teléfono del cliente, fecha y hora). Si la fecha y hora son válidas, se añaden los platos deseados en un bucle, y al salir de este se calcula el precio total, se asigna al pedido el estado solicitado.

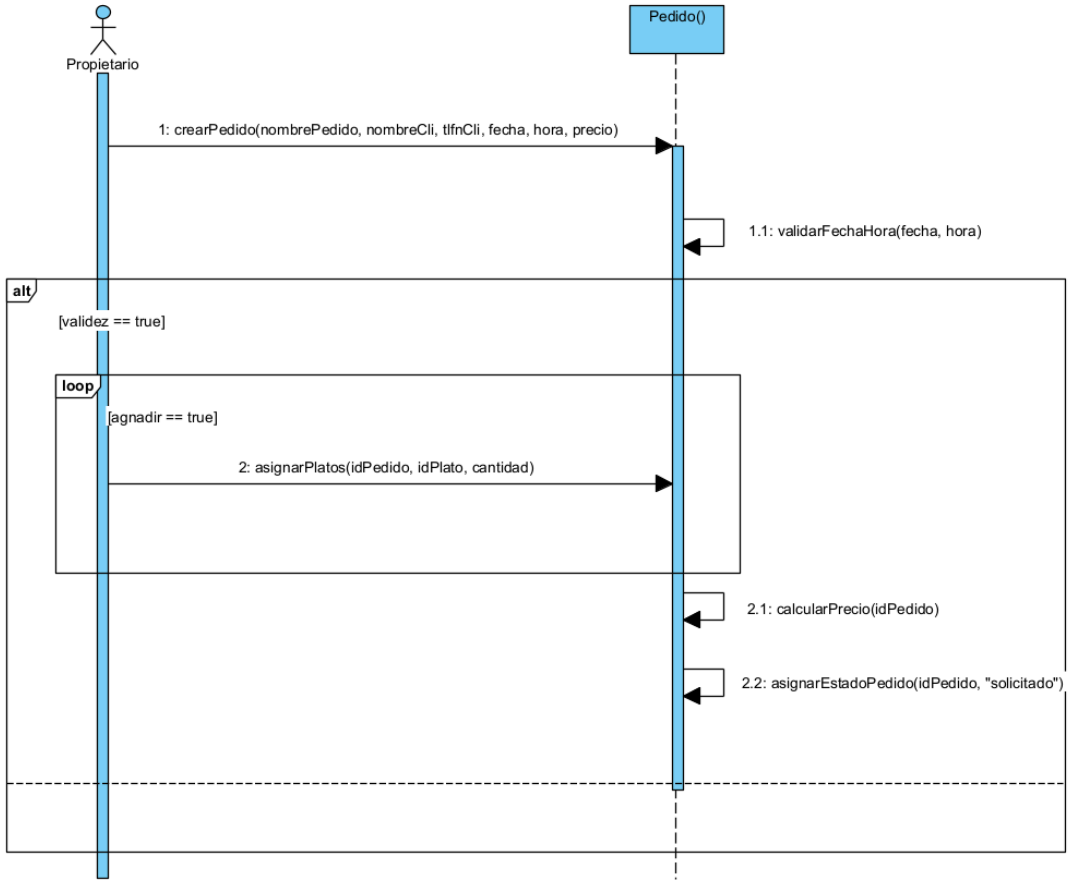


Diagrama de secuencia de modificarPedido

Diagrama bastante similar al de crearPedido, a excepción de que se permite la eliminación de platos pertenecientes a el pedido.

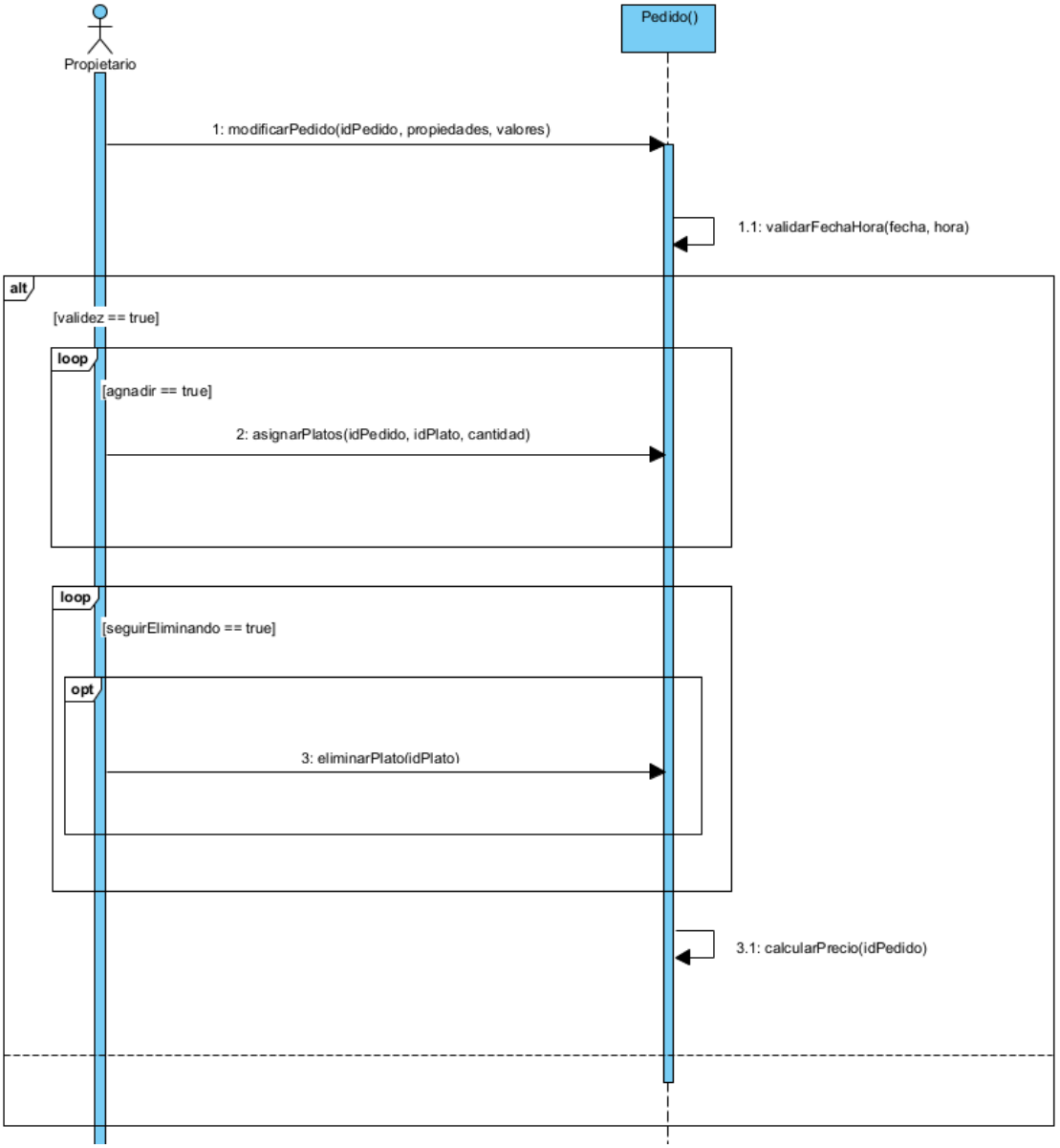


Diagrama de secuencia de eliminarPedido

Diagrama bastante simple, idéntico al de eliminarPlato pero para eliminar un pedido. Tras la llamada a la operación, la línea de vida del pedido muere.

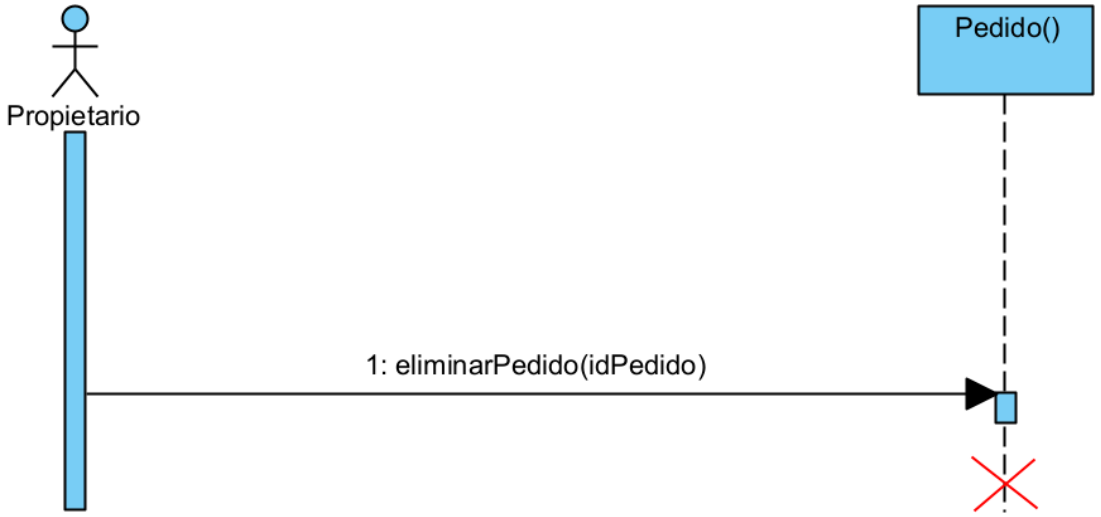
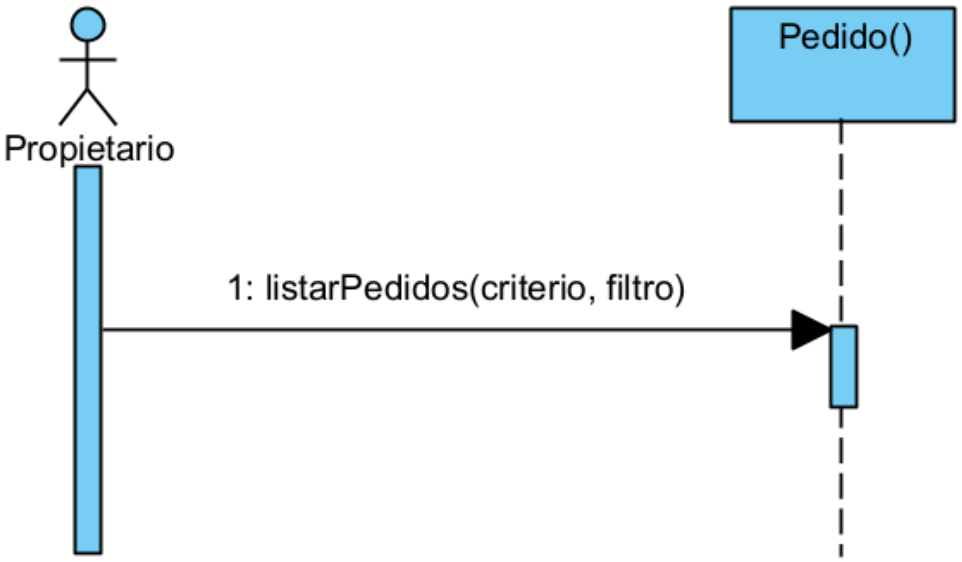


Diagrama de secuencia de listarPedidos

Diagrama en el que el administrador solicita la opción de listar pedidos bajo un determinado criterio de ordenación/filtrado, el cual puede ser nulo.



Prototipos de las pantallas

Para el diseño de los prototipos de las pantallas se ha utilizado la herramienta Visual Paradigm, la cual permite la posibilidad de crear Wireframes para móviles Android.

Se ha optado por un diseño sencillo, intuitivo y atractivo que muestra claramente las distintas opciones de la aplicación. A continuación, se mostrarán los prototipos de pantallas diseñados:

1. Principal pedidos

2. Principal platos

3. Creación de pedido

4. Creación plato

5. Modificación de pedido

6. Modificación de plato

7. Filtrado de pedidos

8. Error fecha inválida

9. Confirmar eliminar pedido

10. Confirmar eliminar plato

11. Ordenación de platos

12. Ordenación de pedidos

13. Confirmación pedido y notificación al cliente

Mapa de navegación

Para crear el mapa de navegación hemos utilizado la herramienta *Wireflow Diagram* de Visual Paradigm. Esta nos permite utilizar los prototipos expuestos anteriormente y relacionarlos.

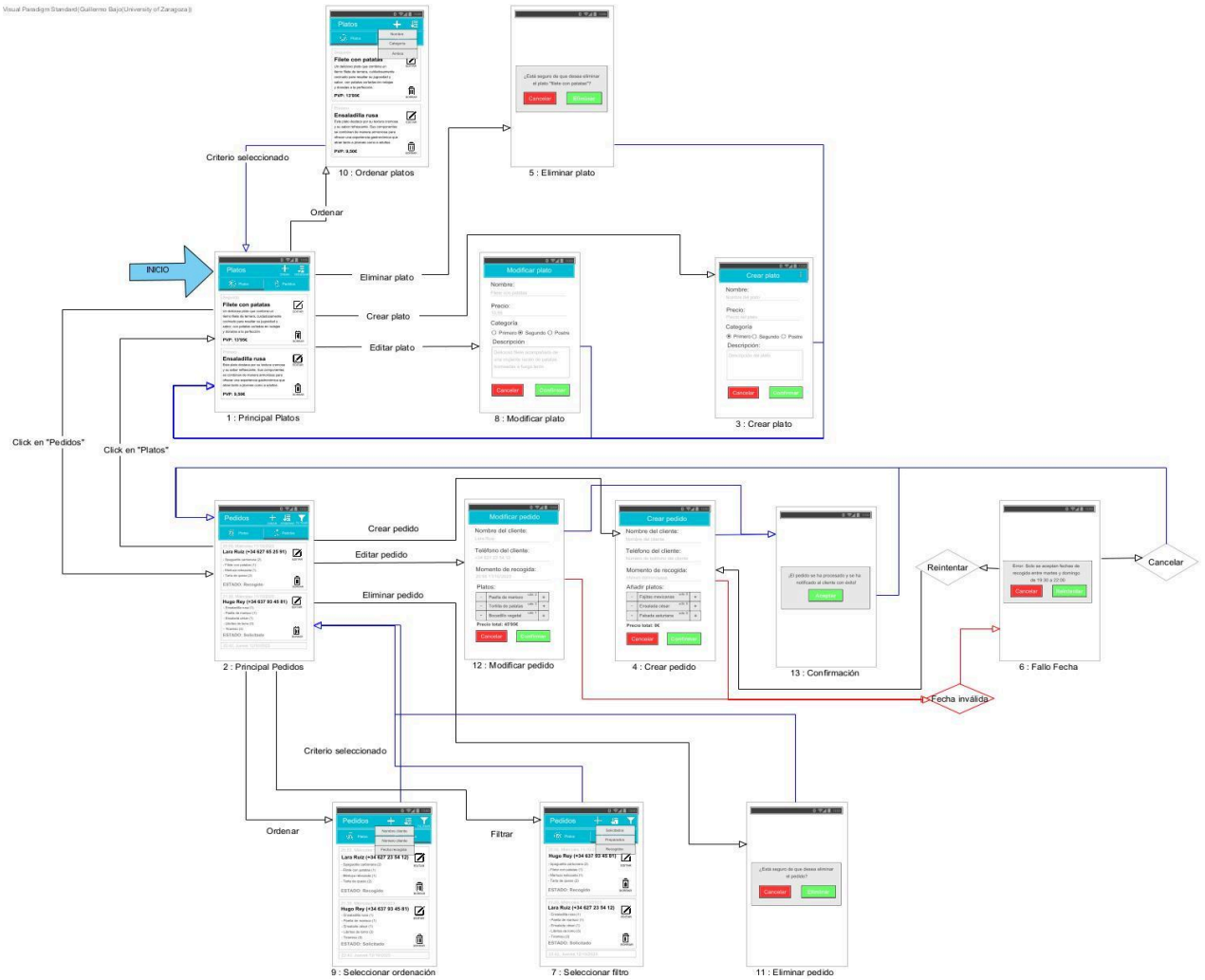
El recorrido empezará siempre en la pantalla *Principal Pedidos*, que nos muestra una lista de los pedidos ordenados por fecha. Desde esta pantalla podremos cambiar a *Principal Platos*, en la que encontraremos un listado de los diferentes platos almacenados en el sistema.

Los flujos de platos y pedidos son casi idénticos, diferenciándose en la posibilidad de filtrar los pedidos por estado (solicitado, preparado, recogido) y en la comprobación que debemos realizar en el campo "momento de recogida", que la fecha introducida se encuentra en el rango en el que el restaurante permite la recogida de pedidos. En este caso podremos cancelar la operación, lo que nos llevará a la pantalla principal, o reintentar la operación, para lo que deberemos poner una fecha válida.

Las operaciones de filtrado y ordenación mostrarán un desplegable con los distintos criterios que podemos aplicar, mostrando el que está seleccionado en ese momento. Una vez seleccionado, volveremos al menú principal en el que los platos o pedidos estarán filtrados y/o ordenados según los criterios seleccionados. Para cambiar o quitar los criterios deberemos volver a hacer click en el icono correspondiente y repetir el proceso.

Las operaciones de creación y modificación son idénticas, cambiando el botón que hay que seleccionar para acceder a cada una de estas y que en la modificación aparecerán los campos ya rellenados.

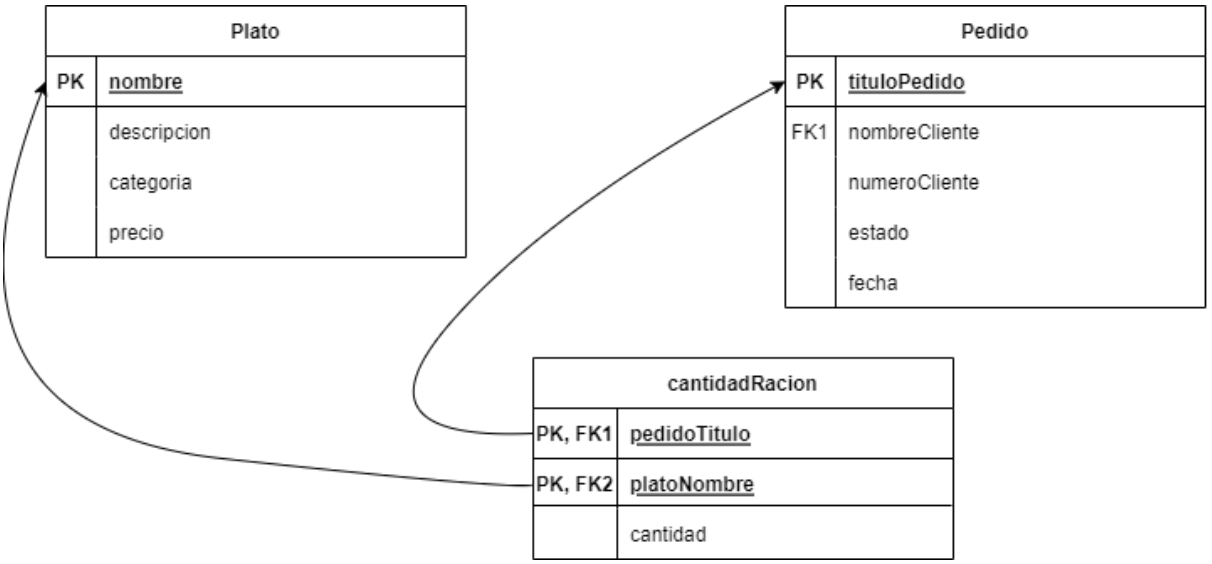
Para eliminar un plato o pedido haremos click en el botón "Eliminar" y confirmaremos la operación. En el caso de que nos hayamos equivocado podremos cancelar la operación.


Visual Paradigm Standard(Guillermo Bajo(University of Zaragoza))
Criterio seleccionado
10 : Ordenar platos
5 : Eliminar plato
Ordenar
INICIO
Eliminar plato
Crear plato
Editar plato
1 : Principal Platos
8 : Modificar plato
3 : Crear plato
Click en "Pedidos"
Click en "Platos"
Crear pedido
Editar pedido
Eliminar pedido
2 : Principal Pedidos
12 : Modificar pedido
4 : Crear pedido
13 : Confirmación
Reintentar
Cancelar
6 : Fallo Fecha
Fecha inválida
Criterio seleccionado
Ordenar
Filtrar
9 : Seleccionar ordenación
7 : Seleccionar filtro
11 : Eliminar pedido

Diseño del sistema y diseño de objetos

Modelo lógico de la base de datos relacional

Para gestionar los datos de nuestra aplicación de gestión de pedidos para una tienda de preparación de comidas se ha procedido a diseñar una base de datos en SQL. Inicialmente, se ha diseñado el modelo relacional, el cual consta de tres entidades (plato, pedido y numRaciones):



Una vez diseñado el modelo relacional, se puede proceder a construir las sentencias SQL de creación de tablas. Se ha creado un fichero crear_bd.sql para la creación de las tablas y otro borrar_bd.sql para el borrado de las mismas, a continuación, el código perteneciente a los ficheros:

crear_bd.sql:

```
CREATE TABLE Plato(
    idPlato      INTEGER(10) PRIMARY KEY,
    nombre       VARCHAR(120),
    descripcion  VARCHAR(255),
    categoria    VARCHAR(60),
    precio       NUMBER(10, 2)
);

CREATE TABLE Pedido(
    idPedido         INTEGER(10) PRIMARY KEY,
    tituloPedido     VARCHAR(120),
    nombreCliente    VARCHAR(120),
    numeroCliente    NUMBER(9),
```

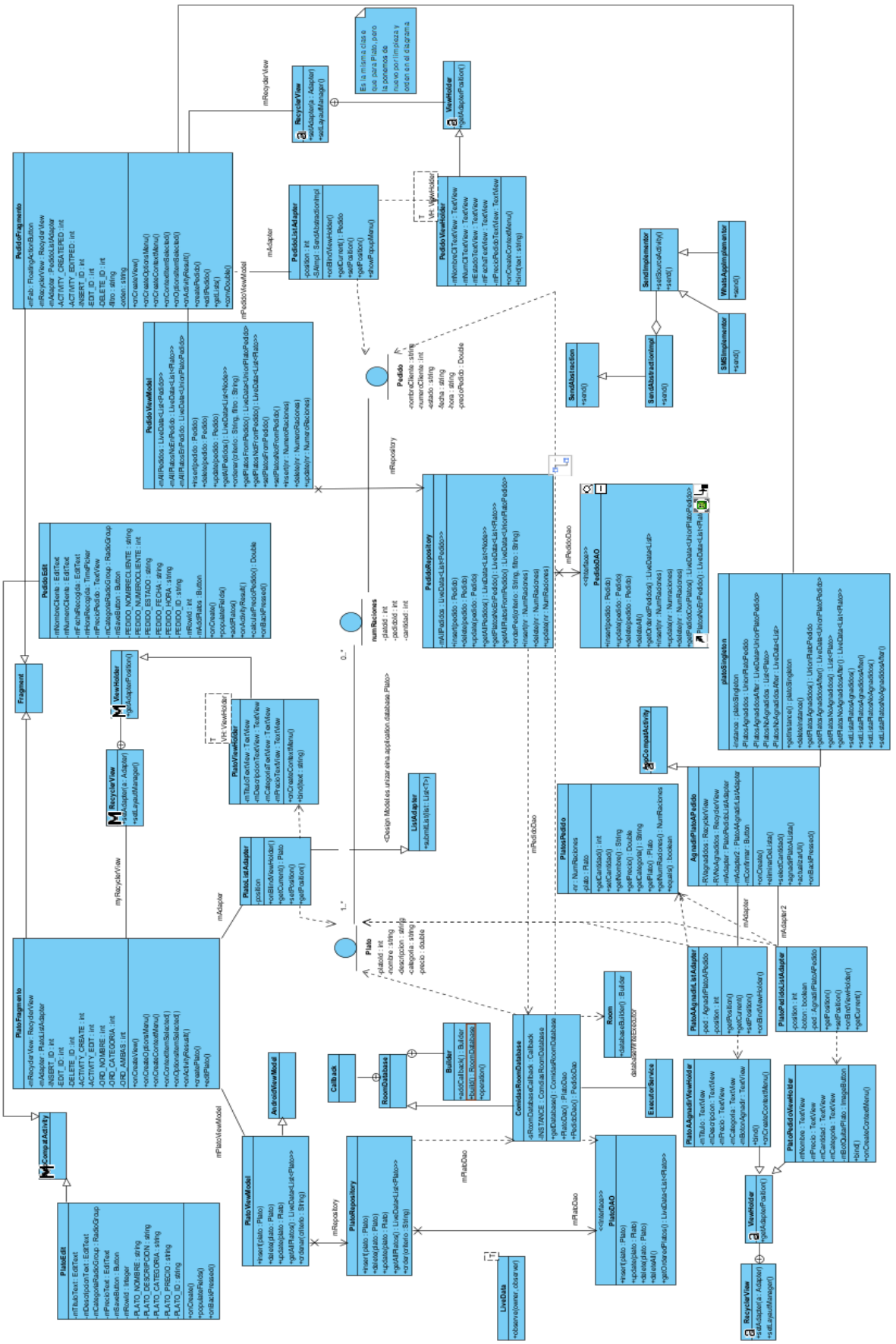
```
    estado          VARCHAR(120),
    fecha           VARCHAR(120),
    hora            VARCHAR(120)

);


CREATE TABLE numRaciones(
    pedidoTit       VARCHAR(120),
    platoNombre     VARCHAR(120),
    cantidad        NUMBER(9),
    FOREIGN KEY(pedidoTit)REFERENCES Pedido(pedidoId)ON DELETE CASCADE,
    FOREIGN KEY (platoNombre)REFERENCES Plato(platoid)ON DELETE CASCADE
);
```

borrar_bd.sql:

```
DROP TABLE numRaciones;
DROP TABLE Pedido;
DROP TABLE Plato;
```

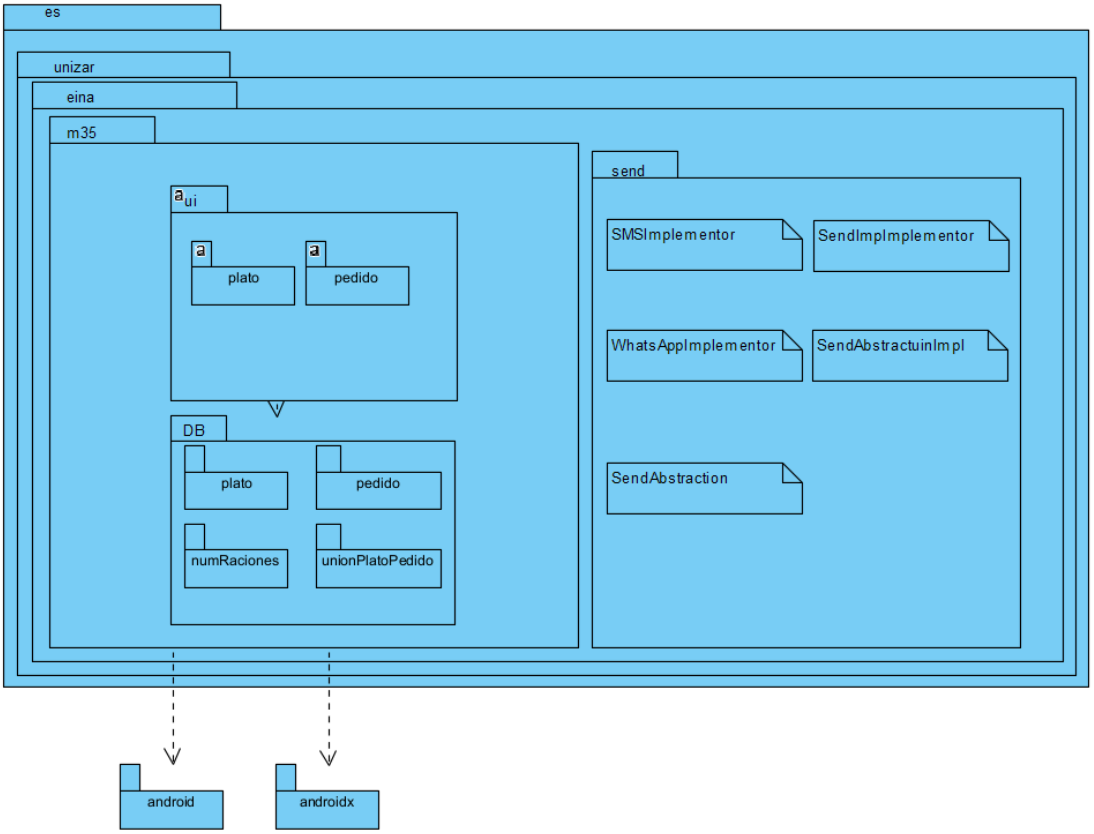
Creación de los diagramas de clase a nivel de diseño

Para la creación del diagrama de clases de diseño se ha tomado como punto de partida el código y diseño visto en la práctica 2 para la gestión de notas, dado la gestión de estas es bastante similar a la de los platos (y pedidos). A continuación, se muestra el diagrama de clases correspondiente a la aplicación de gestión de pedidos para una tienda de preparación de comidas. Dado a su gran tamaño,se va a mostrar en dos imágenes distintas para que sea más legible.



Construcción del diagrama de paquetes

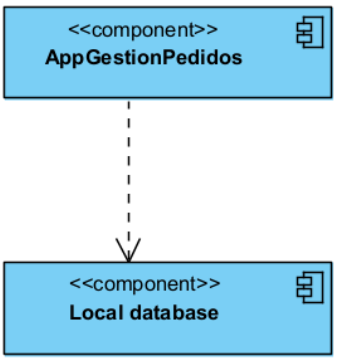
Para reflejar la descomposición en subsistemas de la aplicación de gestión de pedidos para la tienda de preparación de comidas, se ha construido un diagrama de paquetes:



Construcción del diagrama de componentes

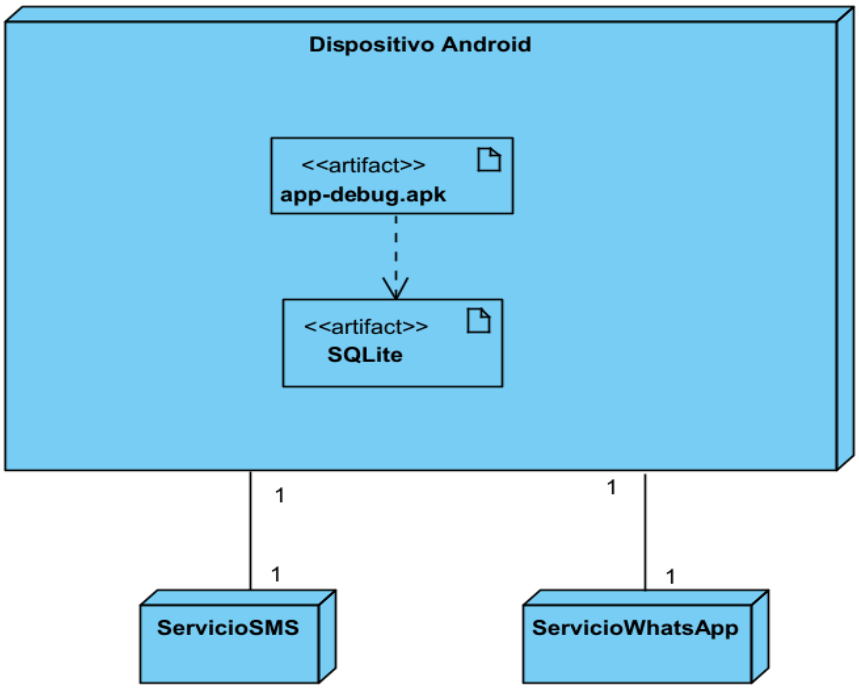
Un diagrama de componentes es una representación visual que muestra la estructura y relaciones entre los componentes de un sistema de software o hardware. Se utiliza para visualizar la arquitectura, comunicar la estructura del sistema, planificar la implementación, identificar dependencias y acoplamientos, y facilitar el mantenimiento y la evolución del sistema.

A continuación, se mostrará el diagrama de componentes correspondiente a nuestra aplicación de gestión de pedidos para la tienda de preparación de comidas:



Construcción del diagrama de despliegue

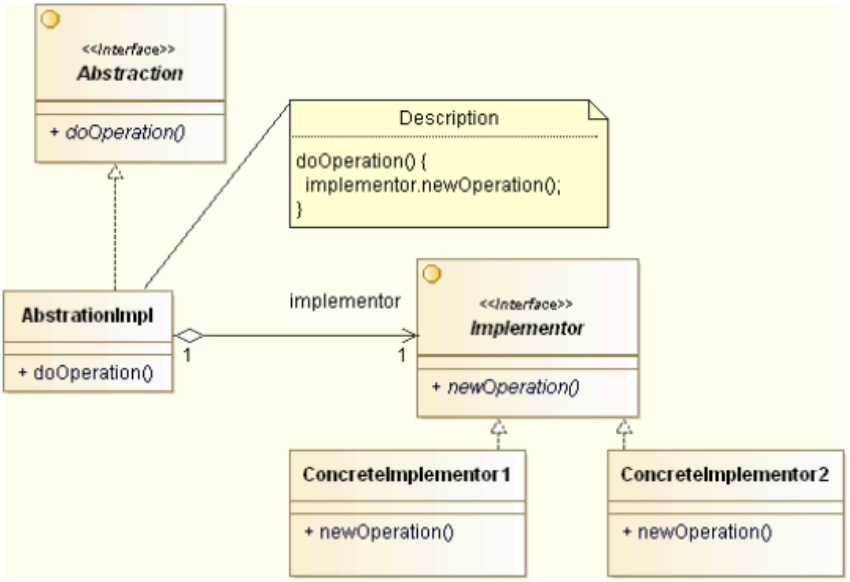
También se ha diseñado un diagrama de despliegue, el cual muestra cómo los componentes de un sistema se distribuyen en el hardware, ayudando a comprender la configuración y conexiones en el entorno físico.



Selección de patrones de diseño

Para desarrollar la funcionalidad de enviar un pedido a un cliente se ha aplicado el patrón de diseño Bridge. Un patrón de diseño proporciona una solución, en forma de plantilla, a un problema que ocurre repetidamente en el desarrollo de software. En particular, el patrón Bridge permite desacoplar una abstracción de su implementación para que las dos puedan variar independientemente.

En la siguiente figura se muestra la estructura del patrón:



- La interfaz *Abstraction* define la interfaz de la abstracción.
- La clase *AbstractionImpl* implementa la interfaz de la abstracción utilizando (delegando en) una referencia a un objeto de tipo *Implementor*.
- La interfaz Implementor define la interfaz para las clases de la implementación. La interfaz no tiene por qué corresponder directamente con la interfaz de la abstracción.
- Las clases *ConcreteImplementor1*, *ConcreteImplementor2*, etc., implementan la interfaz *Implementor*

Construcción de diagramas de secuencia

Finalmente, para representar el modelado dinámico se han construido los diagramas de secuencia a nivel de diseño que permiten representar las trazas de mensajes necesarias para los flujos principales y alternativos planteados en los casos de uso especificados previamente.

Diagrama de secuencia de notificar

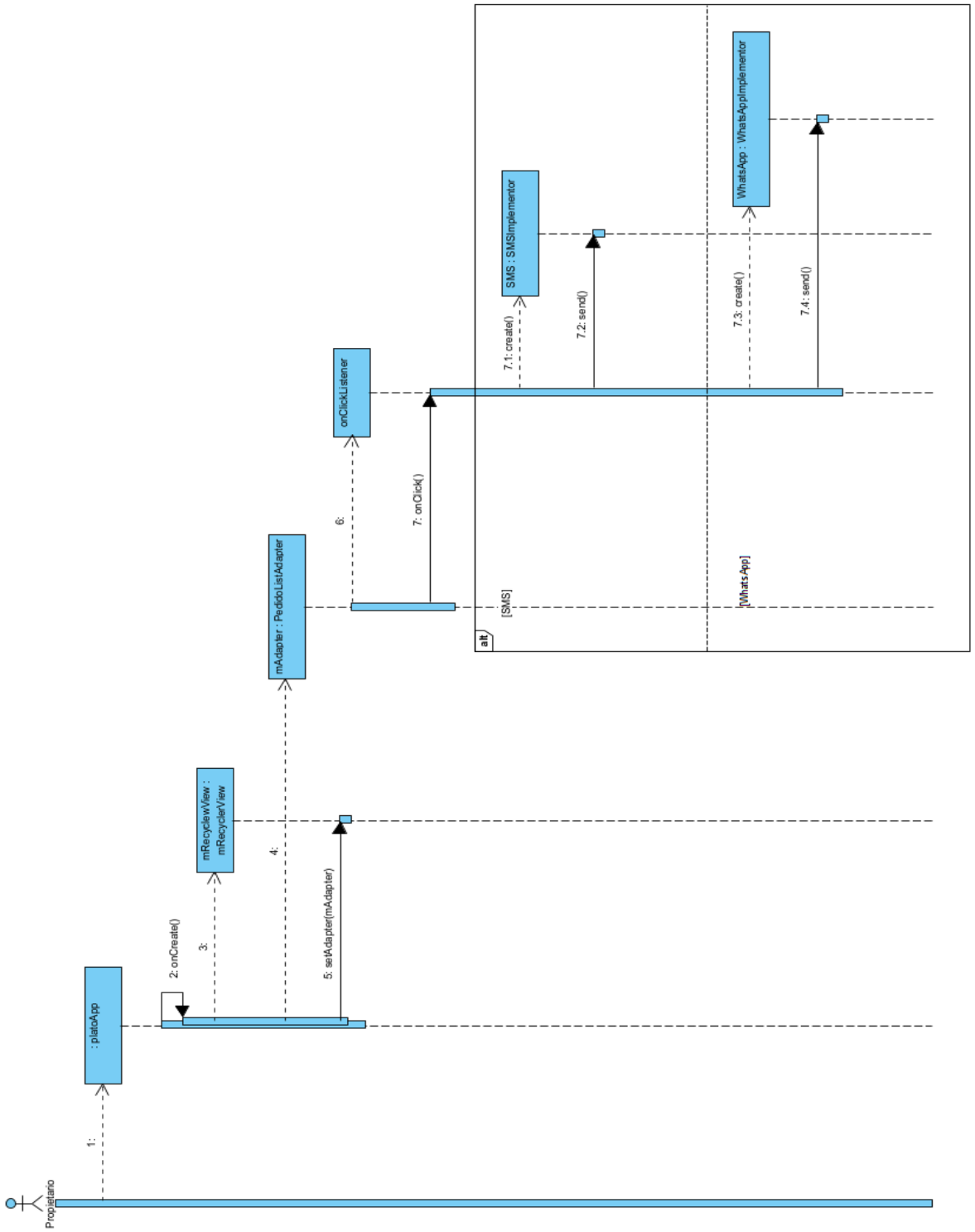


Diagrama de secuencia de crear plato

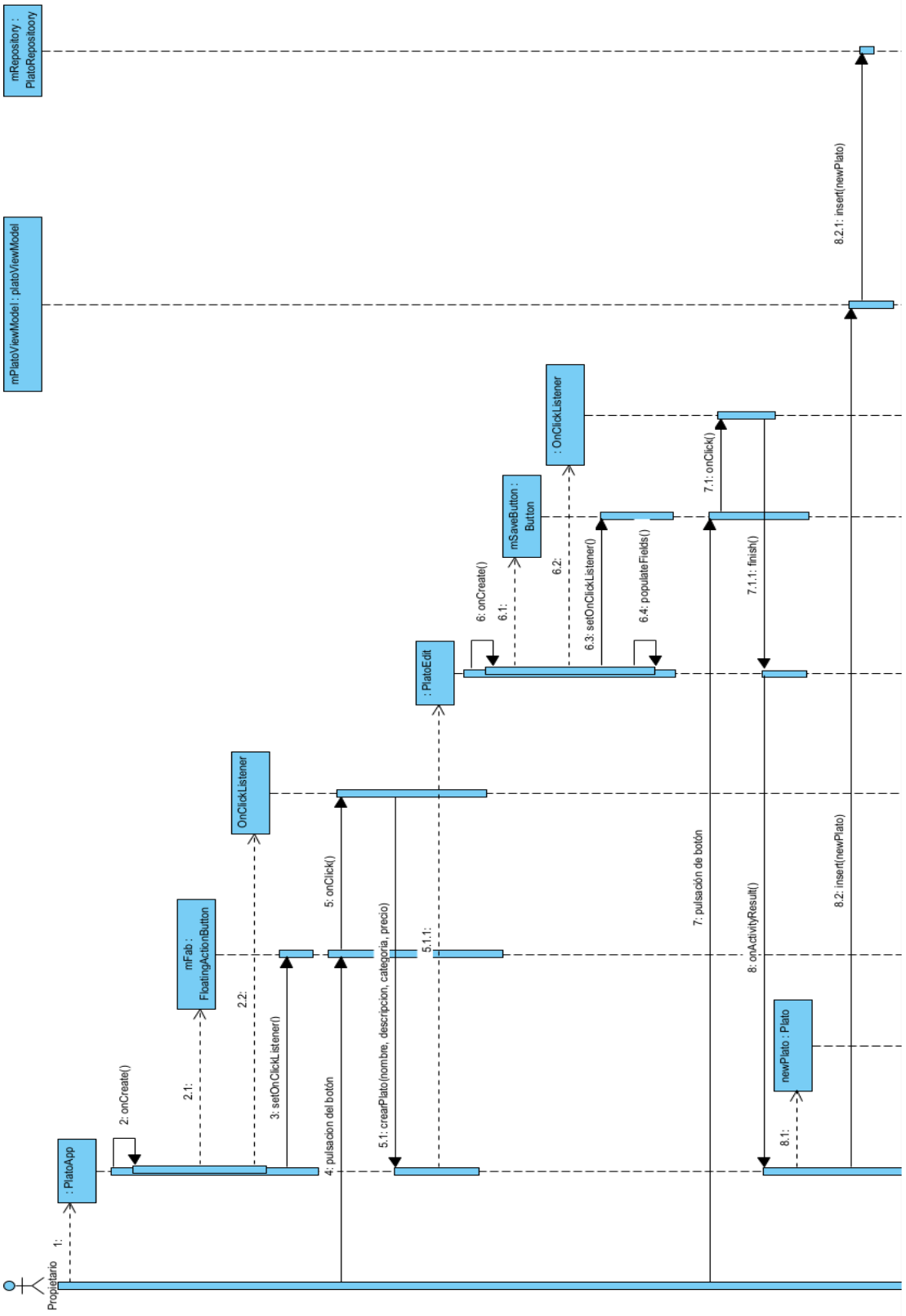


Diagrama de secuencia de modificar plato

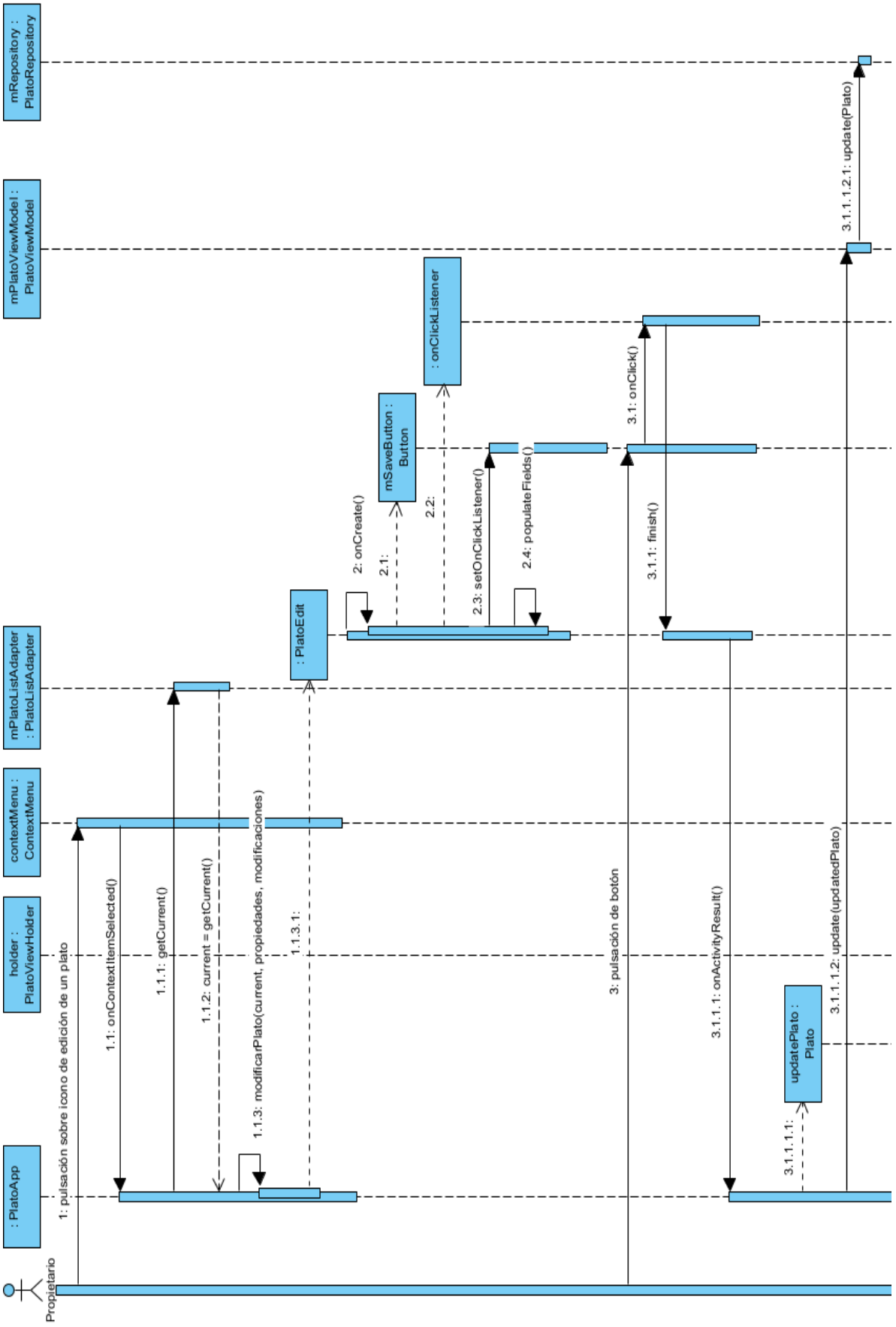


Diagrama de secuencia de eliminar plato

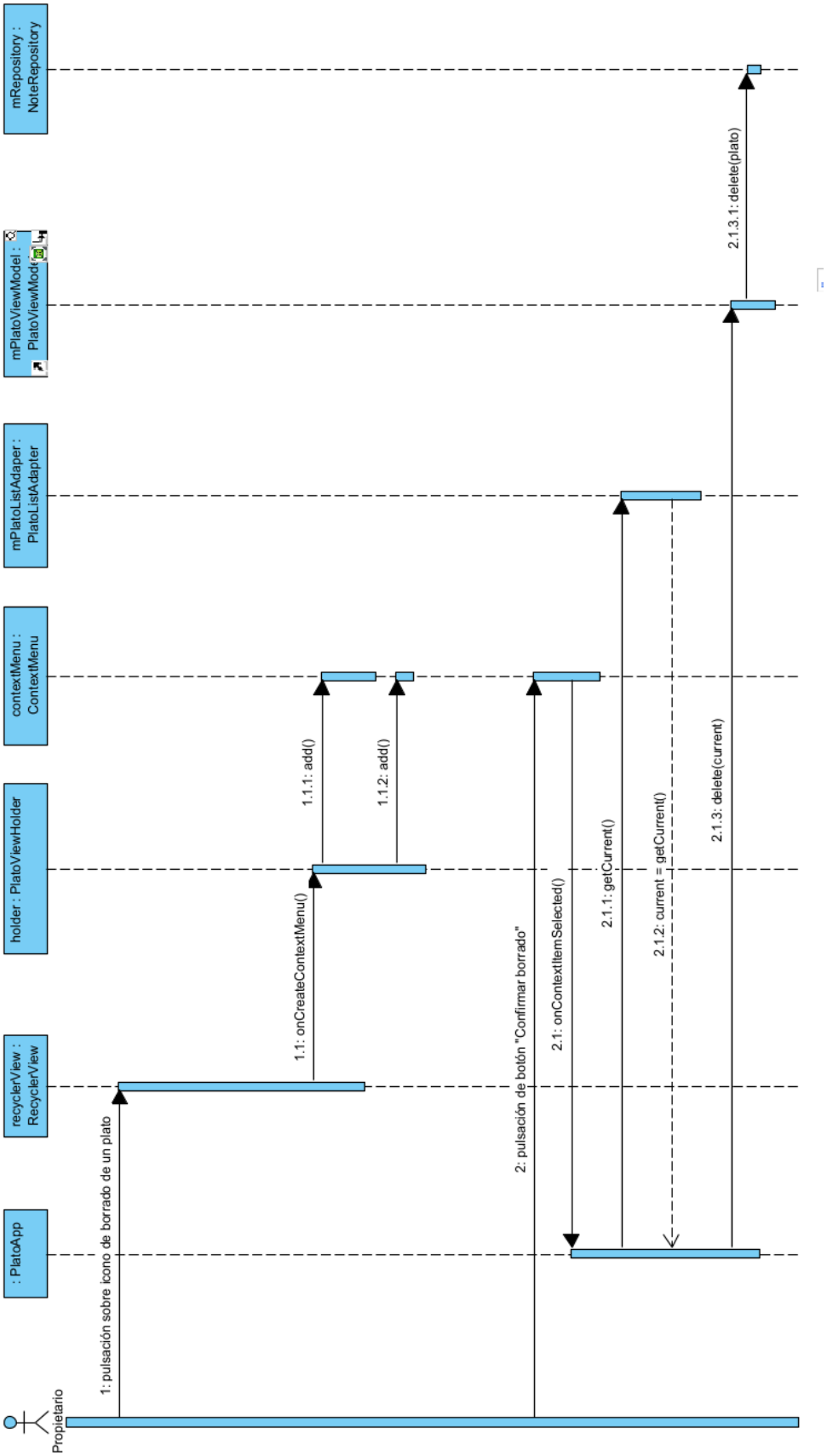


Diagrama de secuencia de listar platos

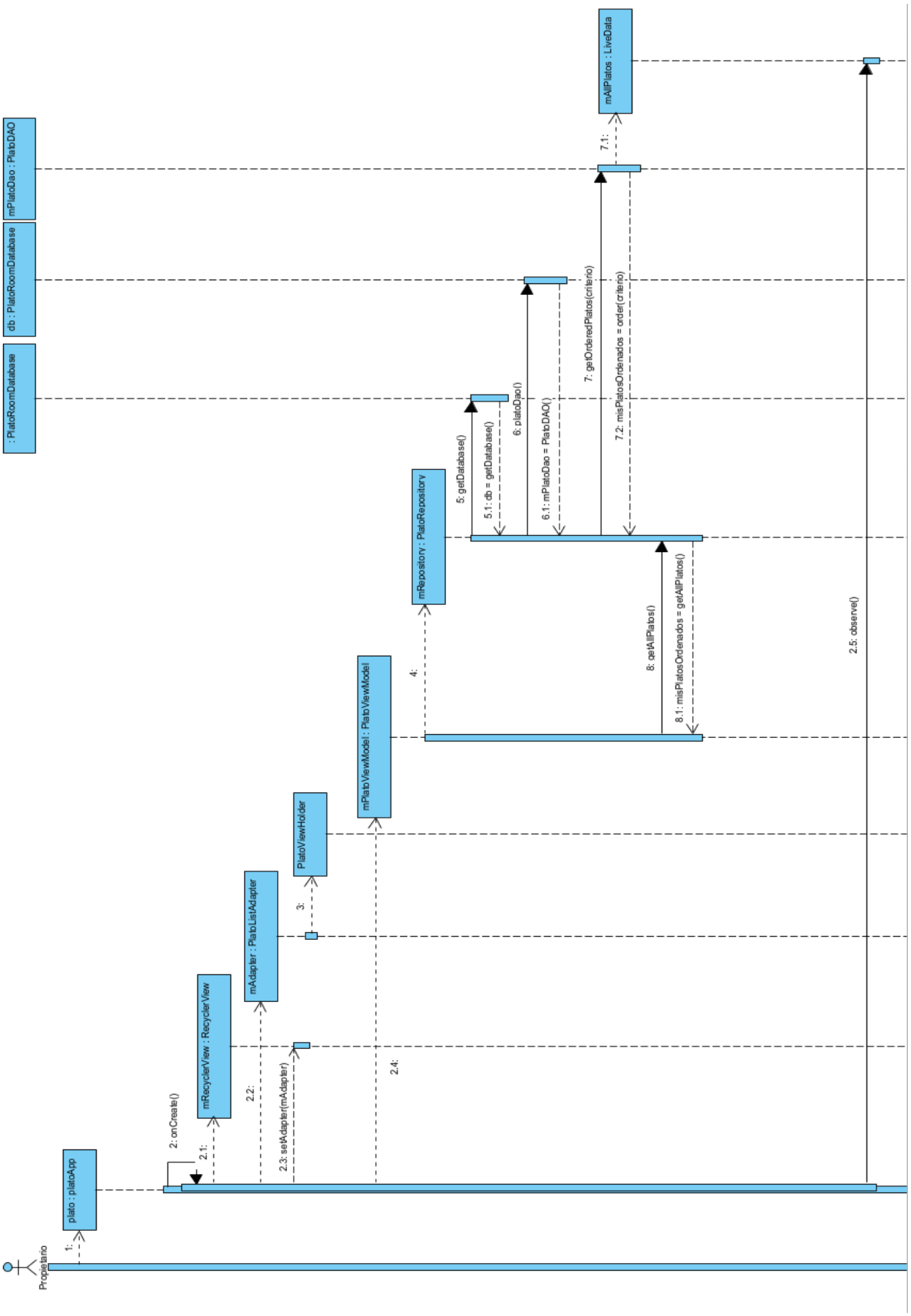


Diagrama de secuencia de crear pedido

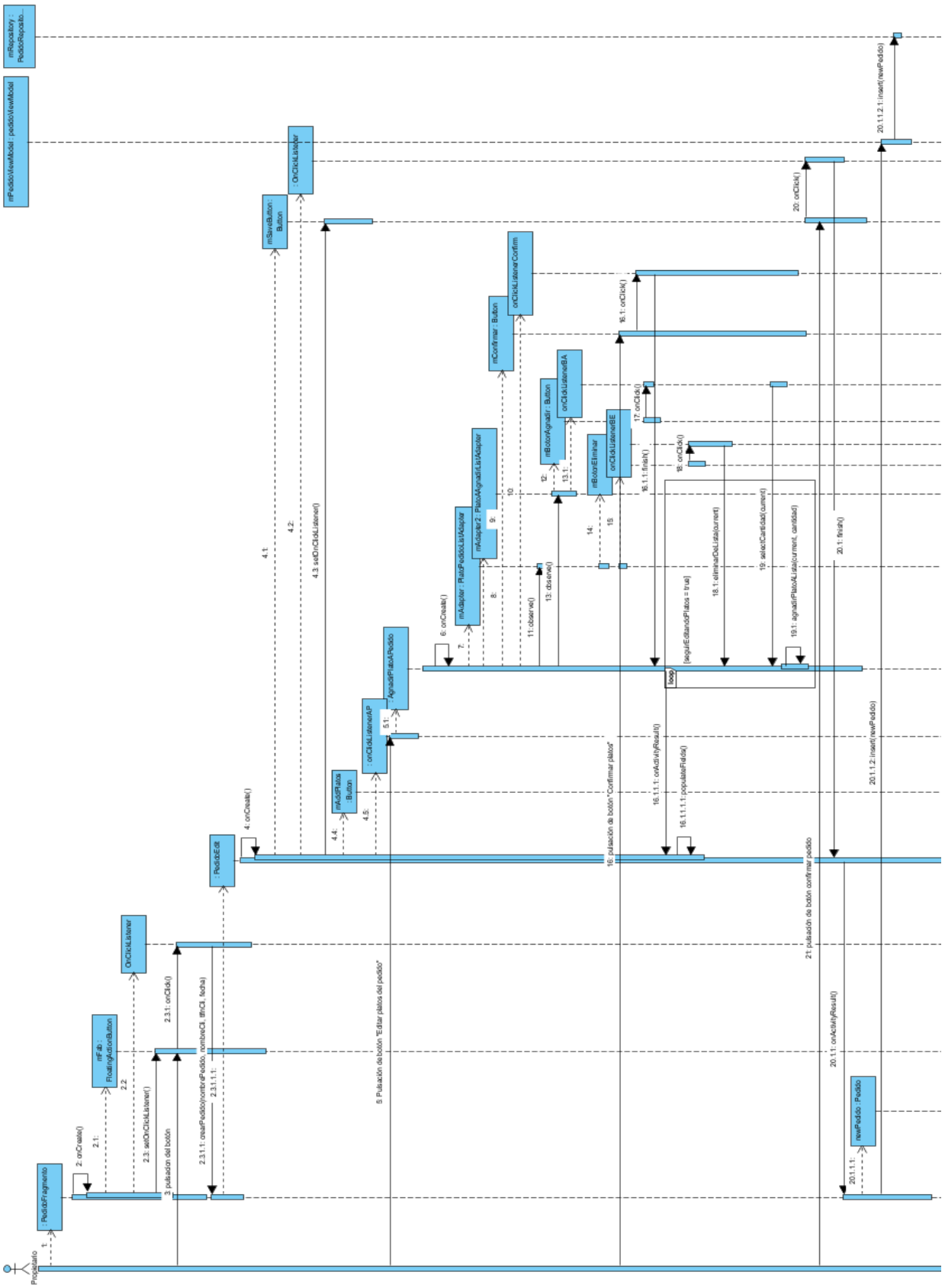


Diagrama de secuencia de modificar pedido

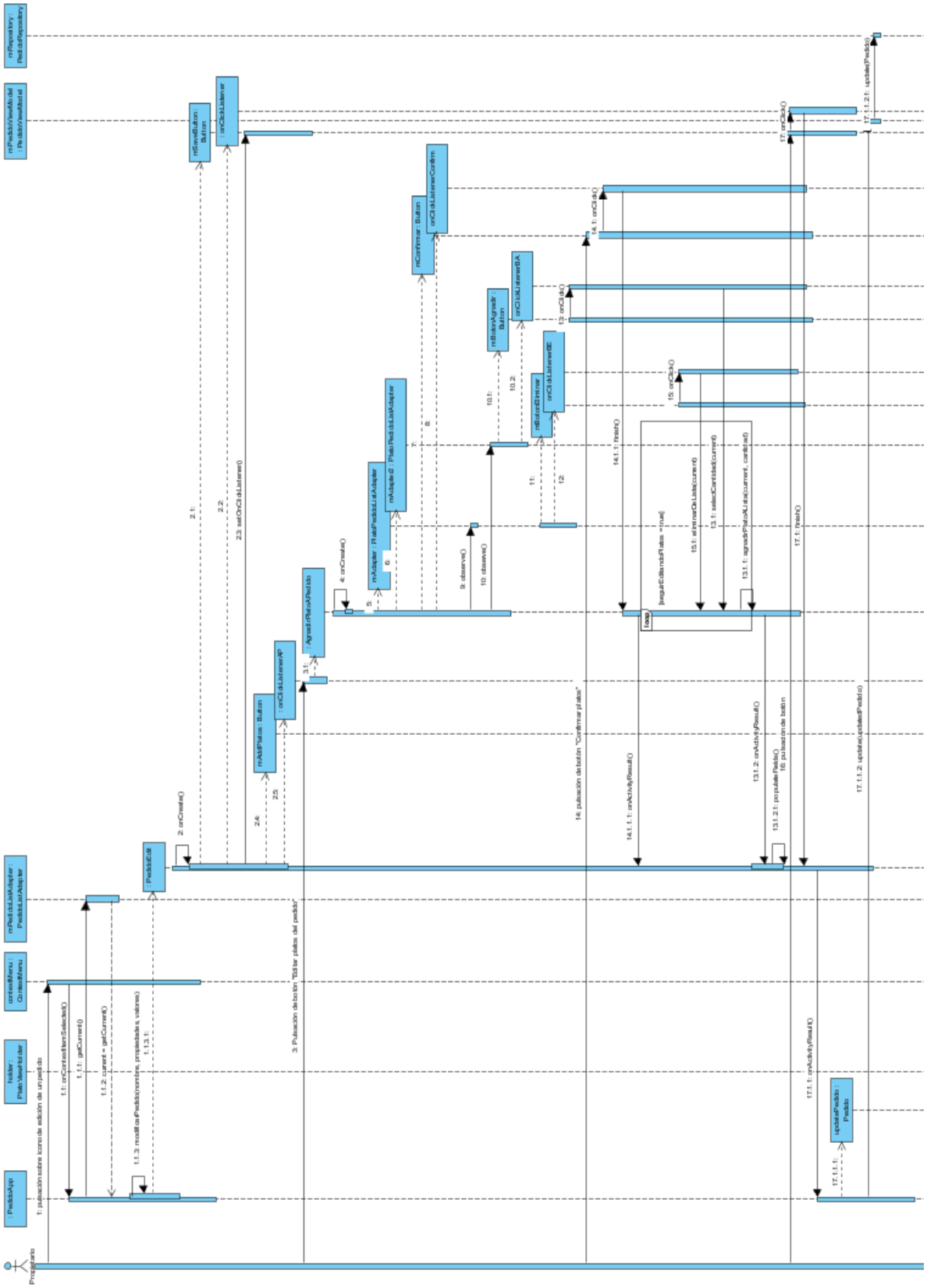


Diagrama de secuencia de eliminar pedido

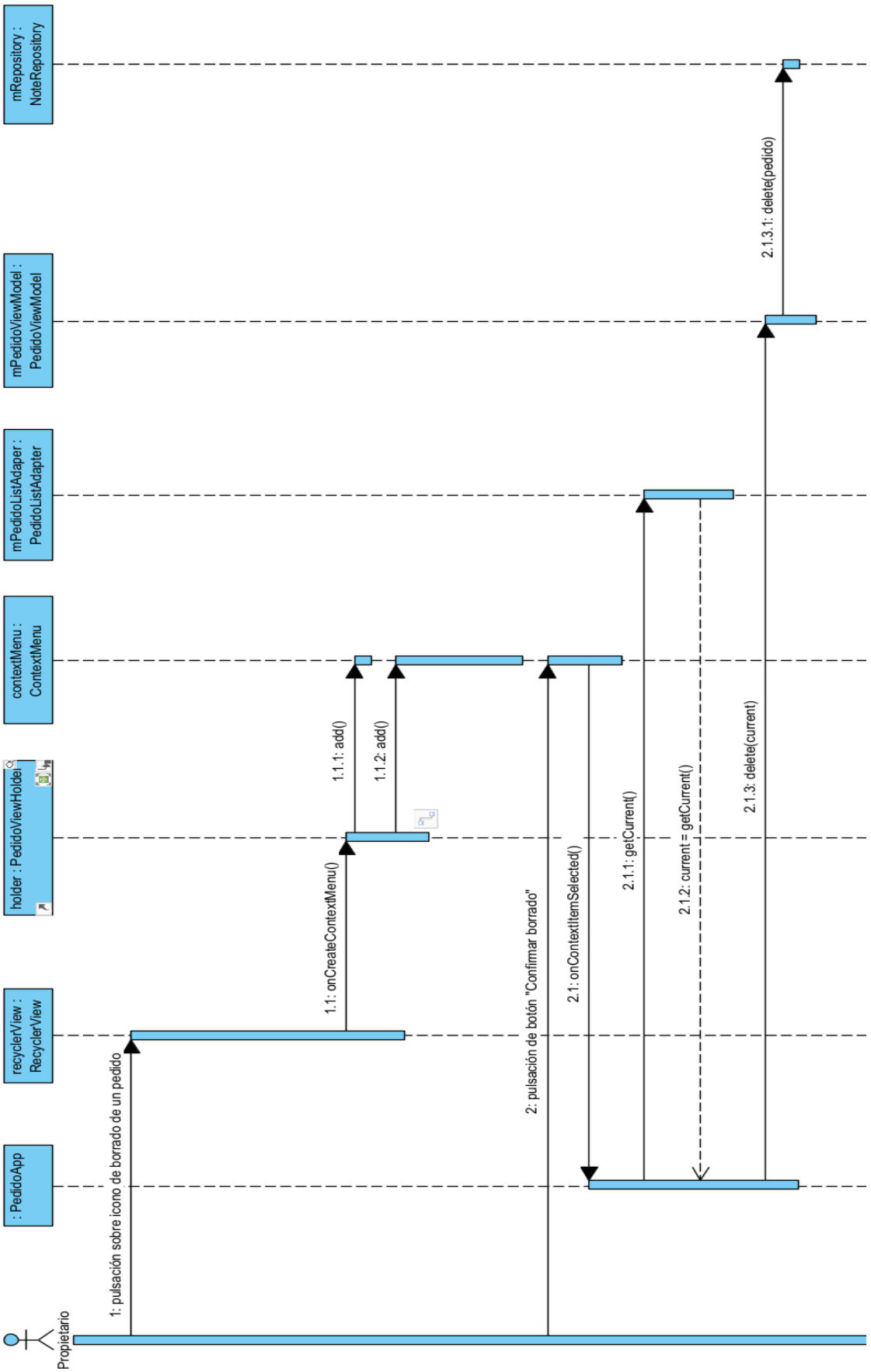
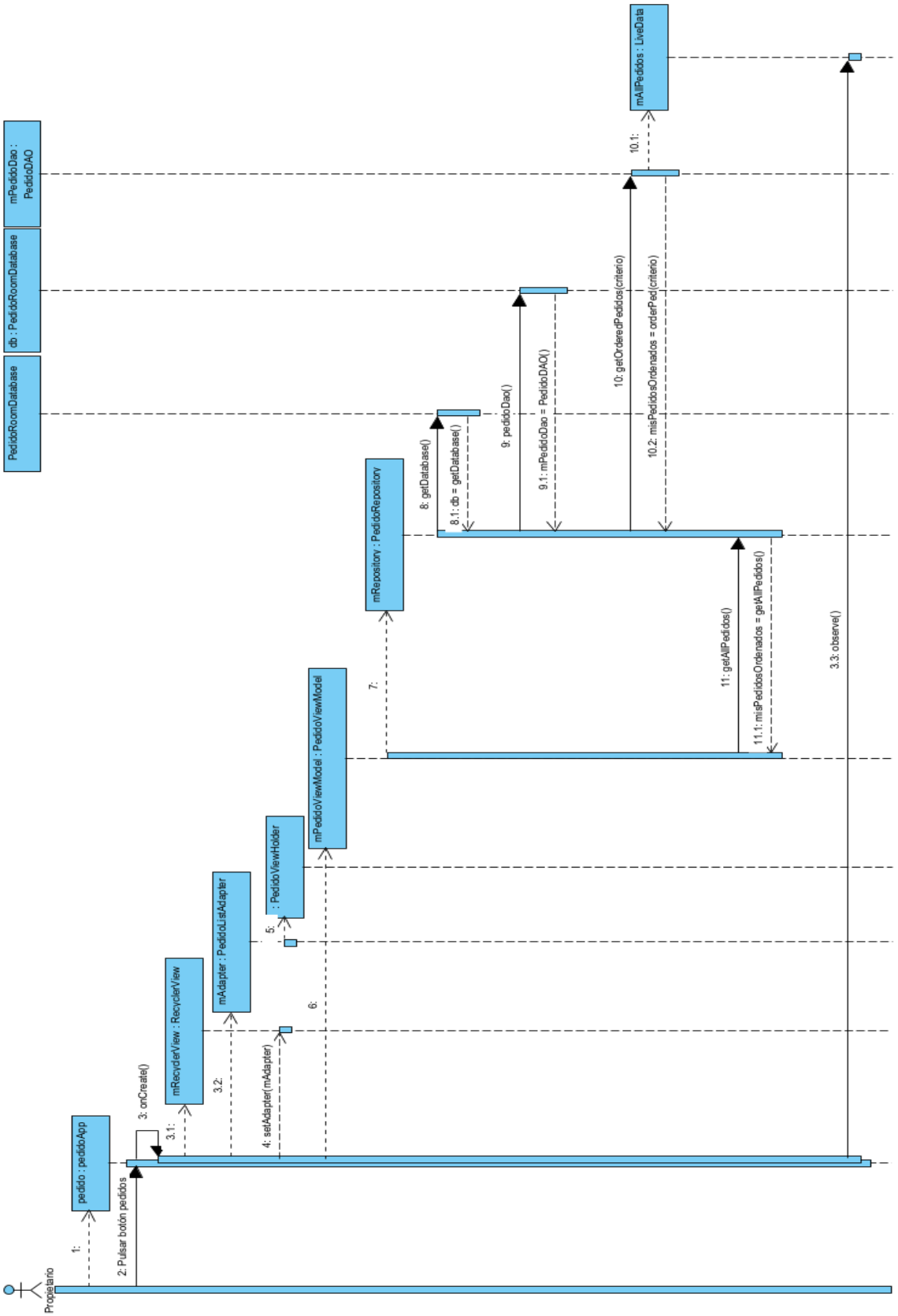


Diagrama de secuencia de listar pedidos



Pruebas

En el marco de esta práctica, se aborda el crucial proceso de diseño, implementación y ejecución de casos de pruebas destinados a identificar posibles errores en la aplicación de gestión de pedidos para la tienda de preparación de comidas.
El propósito fundamental de esta actividad es garantizar la calidad del software mediante la aplicación de diversos tipos de pruebas:

- **Pruebas Unitarias:** Se enfocan en validar el correcto funcionamiento de unidades específicas de código, como funciones o métodos individuales. Estas pruebas permiten verificar la precisión y exactitud del comportamiento esperado de componentes aislados.
- **Pruebas de Sistema de Volumen:** Dirigidas a evaluar el desempeño y la estabilidad del sistema bajo condiciones de carga y volumen de datos normales. El objetivo es asegurar que la aplicación responda de manera óptima y eficiente ante situaciones de uso habitual.
- **Pruebas de Sistema de Sobrecarga:** Enfocadas en someter la aplicación a situaciones extremas de carga, superando los límites normales de operación. Estas pruebas evalúan la capacidad de la aplicación para mantener su funcionalidad y rendimiento bajo presión y picos de demanda.

Pruebas unitarias de caja negra

Comenzamos definiendo las clases de equivalencia para plato y pedido.
El lenguaje que estamos utilizando, Java, ya se encarga de parte de estas verificaciones. Al ser un lenguaje tipado no podemos poner *strings* en variables tipo *int*, por ejemplo. Además, el identificador es asignado por la base de datos en la creación del elemento, por lo que solo tendremos que comprobar el identificador en modificación y eliminación.

Clase "Plato"

Campo	Clase de equivalencia válida	Clase de equivalencia no válida
Id	1) Id > 0 2) $\forall$ i, Id[i] $\in$ [0-9]	3) Id == null 4) Id <= 0 5) $\exists$ i, Id[i] $\notin$ [0-9]
Título	6) Título == [a-zA-Z0-9/s]+ 7) Título.length > 0	8) Título == null 9) Titulo.length == 0
Descripción	10) Descripción == [a-zA-Z0-9]+ 11) Descripción.length > 0	12) Descripción == null 13) Descripción.length == 0
Categoría	14) Categoría $\in$ ("Primero", "Segundo", "Tercero")	15) Categoría == null 16) Categoría $\notin$ ("Primero", "Segundo", "Tercero")
Precio	17) Precio > 0 18) $\forall$ i, Precio[i] $\in$ [0-9]	19) Precio <= 0 20) Precio == null 21) $\exists$ i, Precio[i] $\notin$ [0-9]

Prueba para la creación de un plato

Entrada				**Resul.**	**Clases cubiertas**
Título	**Descripción**	**Categoría**	**Precio**		
Sopa	Con tomate	Primero	10	1	1,2,6,7,10,11,14, 17,18
	Con tomate	Primero	10	-1	9
null	Con tomate	Primero	10	-1	8
Sopa		Primero	10	-1	13
Sopa	*null*	Primero	10	-1	12
Sopa	Con tomate	*null*	10	-1	15
Sopa	Con tomate	NOvalido	10	-1	16
Sopa	Con tomate	Primero	-20	-1	19
Sopa	Con tomate	Primero	*null*	-1	20

Prueba para la modificación de un plato

Entrada					**Resul.**	**Clases cubiertas**
Id	**Título**	**Descripción**	**Categoría**	**Precio**		
1	Canelon	Bechamel	Segundo	10	1	1,2,6,7,10,11,14, 17,18
null	Canelon	Bechamel	Segundo	10	-1	3
-1	Canelon	Bechamel	Segundo	10	-1	4
1		Bechamel	Segundo	10	-1	9
1	*null*	Bechamel	Segundo	10	-1	8
1	Canelon		Segundo	10	-1	13
1	Canelon	*null*	Segundo	10	-1	12
1	Canelon	Bechamel	*null*	10	-1	15
1	Canelon	Bechamel	NOvalido	10	-1	16
1	Canelon	Bechamel	Segundo	-20	-1	19
1	Canelon	Bechamel	Segundo	null	-1	20

Prueba para la eliminación de un plato

Entrada					Resul.	Clases cubiertas
Id	**Título**	**Descripción**	**Categoría**	**Precio**		
1	Sopa	Con tomate	Primero	10	1	1,2,6,7,10,11,14, 17,18
-1	Sopa	Con tomate	Primero	10	-1	4

Clase "Pedido"

Campo	Clase de equivalencia válida	Clase de equivalencia no válida
PedidoId	1) PedidoId > 0 2) ∀ i, PedidoId[i] ∈ [0-9]	3) PedidoId == null 4) PedidoId <= 0 5) ∃ i, PedidoId[i] ∉ [0-9]
NombreCliente	6) NombreCliente == [a-zA-Z]+ 7) NombreCliente.length > 0	8) NombreCliente== null 9) NombreCliente.length = 0
NumeroCliente	10) Nombre != null 11) Nombre.length > 0	12) NumeroCliente == null 13) NumeroCliente <= 0 14) ∃ i, NumeroCliente [i] ∉ [0-9]
Estado	15) Estado ∈ ("Solicitado", "Preparado", "Recogido")	16) Estado == null 17) Estado ∉ ("Solicitado", "Preparado", "Recogido")
Fecha	18) Fecha tiene el formato DD/MM/AAAA	19) Fecha == null 20) Fecha no tiene el formato DD/MM/AAAA
Hora	21) Hora tiene el formato HH:MM	22) Hora == null 23) Hora no tiene el formato HH:MM
PrecioPedido	24) PrecioPedido > 0 25) ∀ i, PrecioPedido [i] ∈ [0-9]	26) PrecioPedido == null 27) PrecioPedido <= 0 28) ∃ i, PrecioPedido [i] ∉ [0-9]

Prueba para la creación de un pedido

Entrada						Resul	Clases cubiertas
Nombre Cli	**Numero Cli**	**Estado**	**Fecha**	**Hora**	**Precio**		
Dean	1	Solicitado	01/12/2023	21:00	10	1	1,2,6,7,10, 11,15,18, 21,24,25

	1	Solicitado	01/12/2023	21:00	10	-1	9
null	1	Solicitado	01/12/2023	21:00	10	-1	8
Dean	*null*	Solicitado	01/12/2023	21:00	10	-1	12
Dean	-1	Solicitado	01/12/2023	21:00	10	-1	13
Dean	1	*null*	01/12/2023	21:00	10	-1	16
Dean	1	NOvalido	01/12/2023	21:00	10	-1	17
Dean	1	Solicitado	*null*	21:00	10	-1	19
Dean	1	Solicitado	NOvalido	21:00	10	-1	20
Dean	1	Solicitado	01/12/2023	*null*	10	-1	22
Dean	1	Solicitado	01/12/2023	NOvalido	10	-1	23
Dean	1	Solicitado	01/12/2023	21:00	*null*	-1	26
Dean	1	Solicitado	01/12/2023	21:00	NOvalido	-1	27

Prueba para la modificación de un pedido

Entrada							**Resul**	**Clases cubiertas**
Id	**Nombre Cli**	**Numero Cli**	**Estado**	**Fecha**	**Hora**	**Precio**		
2	Joank	2	Recogido	01/12/2023	20:30	5	1	1,2,6,7, 10,11,15, 18,21,24, 25
-2	Joank	2	Recogido	01/12/2023	20:30	5	-1	4
null	Joank	2	Recogido	01/12/2023	20:30	5	-1	3
2	*null*	2	Recogido	01/12/2023	20:30	5	-1	9
2		2	Recogido	01/12/2023	20:30	5	-1	8
2	Joank	*null*	Recogido	01/12/2023	20:30	5	-1	12
2	Joank	-2	Recogido	01/12/2023	20:30	5	-1	13
2	Joank	2	*null*	01/12/2023	20:30	5	-1	16

2	Joank	2	NOvalido	01/12/2023	20:30	5	-1	17
2	Joank	2	Recogido	*null*	20:30	5	-1	19
2	Joank	2	Recogido	NOvalido	20:30	5	-1	20
2	Joank	2	Recogido	01/12/2023	*null*	5	-1	22
2	Joank	2	Recogido	01/12/2023	NOva lido	5	-1	23
2	Joank	2	Recogido	01/12/2023	20:30	*null*	-1	26
2	Joank	2	Recogido	01/12/2023	20:30	NOval ido	-1	27

Prueba de eliminación de un pedido

Entrada							**Resul**	**Clases cubiertas**
Id	**Nombre Cli**	**Numero Cli**	**Estado**	**Fecha**	**Hora**	**Precio**		
2	Joank	2	Recogido	01/12/2023	20:30	5	1	1,2,6,7, 10,11,15, 18,21,24, 25
-2	Joank	2	Recogido	01/12/2023	20:30	5	-1	4

Las conclusiones extraídas de estas pruebas son consistentes con lo esperado inicialmente. Se observó que no se generaron errores visibles durante el proceso. Los platos y los pedidos con valores incorrectos no fueron creados, modificados ni eliminados, lo que indica la solidez y robustez de la aplicación en términos de persistencia de datos.

Este resultado refuerza la confianza en la integridad de la aplicación, demostrando su capacidad para mantener la coherencia y precisión de los datos almacenados, evitando posibles anomalías o inconsistencias en el sistema.

Prueba de volumen

Se ha llevado a cabo una exhaustiva evaluación de la aplicación para asegurar la conformidad con los requisitos establecidos al inicio del proyecto. Dichos requisitos especifican que la aplicación debe mantener un rendimiento estable al manejar hasta 100 platos y 2000 pedidos almacenados.

Se han implementado pruebas que simulan la inserción de platos hasta alcanzar la cantidad de 100 y la creación de pedidos hasta llegar a 2000. Una vez finalizadas estas pruebas, se realiza una exhaustiva verificación para identificar posibles errores en la ejecución o en el

funcionamiento general de la aplicación tras la inserción masiva de datos. Los resultados obtenidos confirman que la aplicación opera de manera óptima incluso después de añadir este volumen de información.

Es importante destacar que la aplicación demuestra un rendimiento estable al realizar operaciones de modificación, eliminación y establecimiento de relaciones entre pedidos y platos, todo ello sin presentar inconvenientes o errores significativos.

Prueba de sobrecarga

Durante las pruebas de estrés, se llevó a cabo un análisis específico del campo de descripción asociado a la entidad de platos. Para este propósito, se implementó un método que, en un ciclo repetitivo, insertaba el mismo plato con una descripción en constante aumento en cada iteración. Dentro de este ciclo, el plato recién insertado se elimina para evitar que el crecimiento en la cantidad de platos afectará las conclusiones de las pruebas.

Durante la ejecución de estas pruebas, se observó que la inserción de la descripción del plato continuaba sin problemas hasta alcanzar 10000000 caracteres, momento en el cual la base de datos alcanzaba su capacidad máxima. Obtenemos un error *OutOfMemory* y la aplicación dejaba de funcionar correctamente.

Resultados y conclusiones

En conclusión, estamos satisfechos con los resultados finales de nuestra aplicación. Consideramos que hemos llevado a cabo un trabajo sólido tanto en el diseño del sistema como en la interfaz, y esto se ha traducido en una experiencia de usuario agradable y sin contratiempos.

Hemos logrado alcanzar las metas establecidas en prácticas anteriores, tanto en términos de implementación como en el diseño de la interfaz, aunque se realizaron modificaciones leves para mejorar la usabilidad.

Admitimos que enfrentamos desafíos al sumergirnos en el desarrollo de aplicaciones para Android, dado que carecíamos de conocimientos previos en este campo. Fue necesario invertir tiempo significativo en investigación para superar esta barrera. Además, durante la implementación de modificaciones en la base de datos, nos enfrentamos a desafíos relacionados con errores de migración en el módulo "ComidasRoomDatabase" debido a cambios de versión. Aunque nos costó llegar a la solución, la eliminación de la caché y reconstrucción soluciona el problema

En una nota positiva, la utilización de Logcat se reveló fundamental para el proceso de depuración, respaldado por las herramientas de debugging disponibles. La experiencia global de desarrollo en Android Studio resultó altamente satisfactoria.

Bibliografía

Patrón Bridge: https://refactoring.guru/es/design-patterns/
Android Studio: https://developer.android.com/

La bibliografía está completada por las diapositivas de teoría suministradas en los distintos recursos de moodle. Así como los distintos enunciados de las prácticas.