TRABAJO INGENIERÍA DE SOFTWARE: AMPLIACION APLICACIÓN DE NOTAS PARA ANDROID

Curso *2019-2020*

Álvaro Echávarri, *737400*
Abel Naya, *544125*

1.-Resumen, introducción y objetivos.

Para el análisis, diseño y prueba de nuestra aplicación de notas para móvil, partimos de un simple esquema con 4 funcionalidades: crear nota, modificar nota, borrar nota y mostrar notas. Tal y como nos mostraron los profesores, la idea base era establecer una base de datos y una aplicación que comunique al usuario con esa base de datos. La base de datos almacena las notas y sus elementos y la aplicación muestra esas notas y las diferentes funcionalidades.

Siguiendo el esquema de prácticas propuesto por el profesorado comenzamos completando un código de prueba de nuestra aplicación, que nos sirvió más tarde para realizar los distintos diagramas de análisis que aparecen en este informe, más adelante completamos una versión de la aplicación algo más compleja, que permite dar un título y un cuerpo a cada nota.

Como último añadido, realizamos diagramas mucho más detallados y extensos que se encuentran incluidos también en este pdf como diagramas o fase de diseño, en estos también se añadió una nueva funcionalidad, las categorías, una nota podía tener una categoría y se podían filtrar por su nombre, la base de datos también se encargaría de almacenar estas categorías. Además añadimos el ordenar la lista de notas por título y por categoría tal y como pedía la entrega de este trabajo.

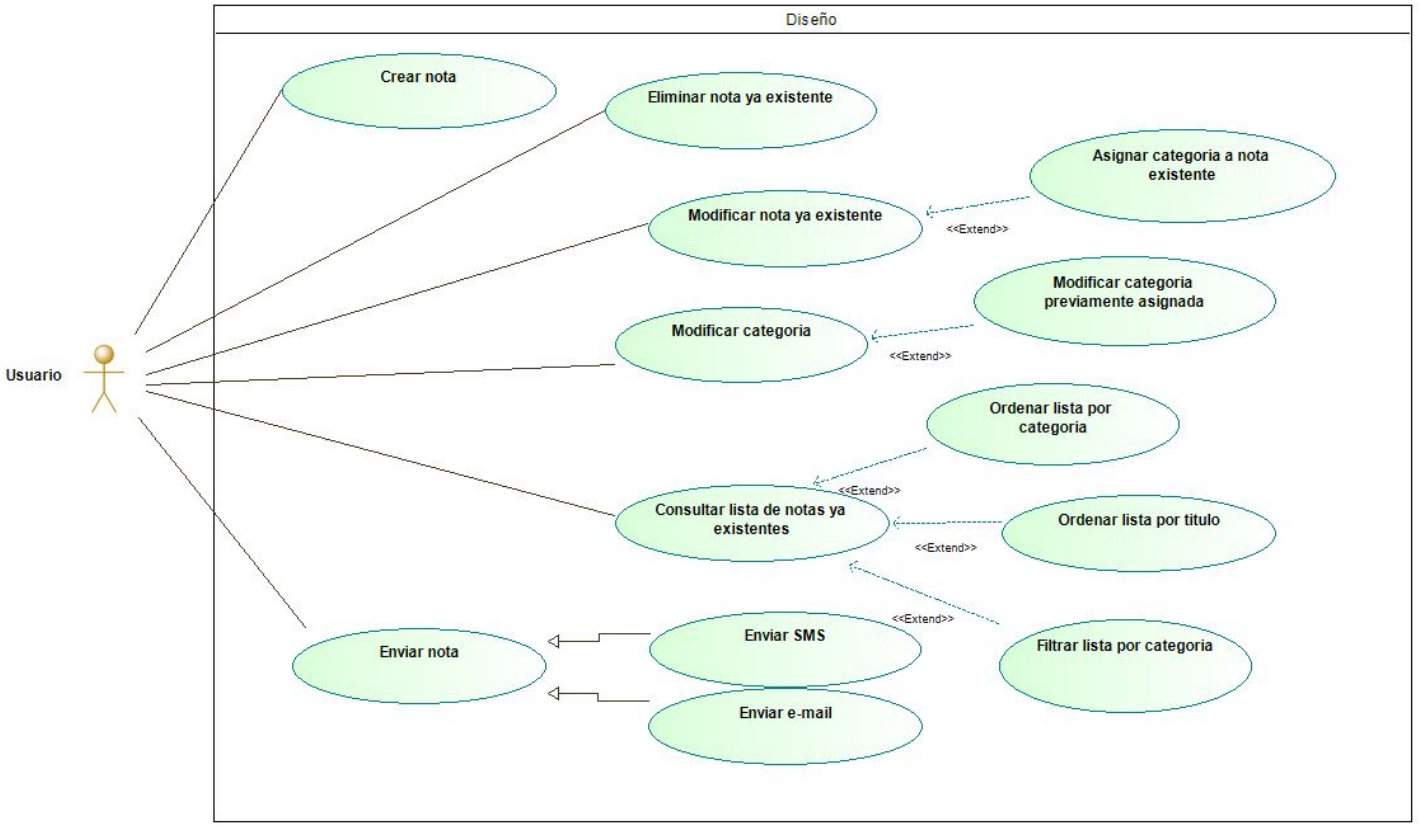
2.-ÍNDICE

3.-Requisitos.

3.1-Catálogo de requisitos.

RF1	El sistema debe tener notas. Cada nota se identificara con un titulo (string), y un cuerpo (string).
RF2	El sistema debe permitir crear notas
RF3	El sistema debe permitir modificar el cuerpo y título de una nota.
RF4	El sistema debe permitir borrar una nota.
RF5	El sistema debe permitir mostrar una lista con todas las notas. No ordenadas de forma predeterminada.
RF6	El sistema debe tener categorías de notas. Cada categoría se identifica por un string.
RF7	El sistema debe permitir crear categorías.
RF8	El sistema debe permitir modificar el identificador de las categorías.
RF9	El sistema debe permitir eliminar categorías.
RF10	El sistema debe permitir asignar una categoría previamente creada a una nota.
RF11	El sistema debe permitir modificar la categoría asignada a una nota.
RF12	El sistema debe permitir eliminar la categoría asignada a una nota.
RF13	El sistema debe permitir ordenar la lista de notas por categoría.
RF14	El sistema debe permitir ordenar la lista de notas por título.
RF15	El sistema debe permitir mostrar sólo las notas de una categoría.
RF16	El sistema debe permitir enviar notas.
RNF 17	El sistema no va a almacenar más de 1000 notas.

3.2-Diagrama de casos de uso



3.2.1-Flujos requisitos funcionales

Crear nota:

-**Flujo principal**

- Inicio en el menú principal
- Desplegar menú.
- Darle a crear.
- Introducir un título y un cuerpo.
- Aceptar

Modificar nota ya existente:

-Flujo principal

- Inicio en el menú principal
- Elegir una nota
- Darle a editar
- Cambiar el título y cuerpo.
- Aceptar

Borrar nota ya existente:

-Flujo principal

- Inicio en el menú principal
- Elegir una nota
- Darle a eliminar.

Consultar lista de notas ya existente:

-**Flujo principal**

- El caso de uso se inicia cuando se arranca la app.
- El sistema recupera las notas y muestra sus títulos.

-**Flujo alternativo:**

- Cuando termina el caso de uso Crear Nota se inicia el caso de uso.
- Caso: Cuando termina el caso de uso Modificar Nota se inicia el caso de uso.
- Cuando termina el caso de uso Eliminar Nota se inicia el caso de uso.
- Cuando termina el caso de uso Eliminar categoría se inicia el caso de uso.
- Cuando termina el caso de uso Crear categoría se inicia el caso de uso.
- Cuando termina el caso de uso Modificar categoría se inicia el caso de uso.
- Cuando termina el caso de uso Asignar categoría se inicia el caso de uso.

Crear categoría:

-**Flujo principal**

- Inicio en el menú principal
- Desplegar menú
- Elegir categorías
- Desplegar menú
- Darle a añadir
- Introducir un nombre
- Aceptar

Modificar categoría:

-Flujo principal

- Inicio en el menú principal
- Desplegar menú
- Elegir categorías
- Elegir una categoría
- Darle a editar
- Cambiar el nombre
- Aceptar

Asignar categoría a nota ya existente:

-Flujo principal

- Inicio en el menú principal
- Elegir una nota
- Darle a editar
- Asignar una categoría.
- Aceptar

Eliminar categoría:

-Flujo principal

- Inicio en el menú principal
- Desplegar menú
- Elegir categorías
- Elegir una categoría
- Darle a eliminar

Ordenar lista de notas por categoría

-Flujo principal

- Inicio en el menú principal
- Elegir ordenar por categoría

Ordenar lista de notas por título

-Flujo principal

- Inicio en el menú principal
- Elegir ordenar por título

Mostrar sólo las notas de una categoría

-Flujo principal

- Inicio en el menú principal
- Elegir categoría en el desplegable
- Marcar opción filtrar por categoría

Enviar notas por E-MAIL

-Flujo principal

- Inicio en el menú principal
- Elegir una nota
- Darle a enviar por e-mail

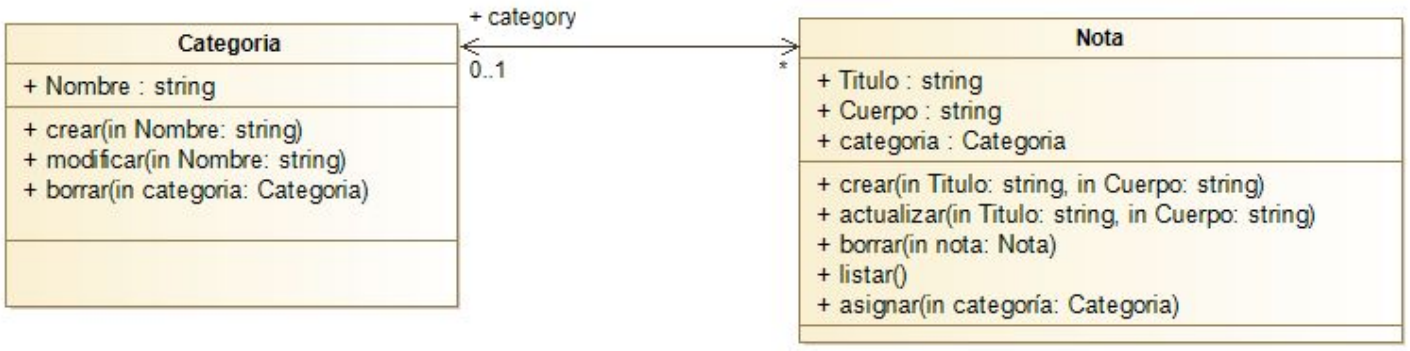
Enviar notas por SMS

-Flujo principal

- Inicio en el menú principal
- Elegir una nota
- Darle a enviar por sms

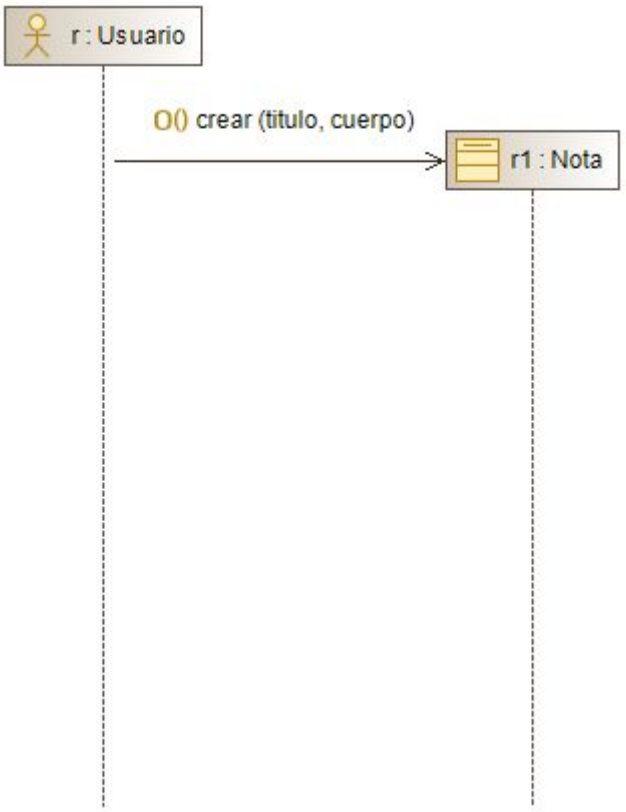
4.-Diagramas fase de análisis

4.1-Diagrama de clases

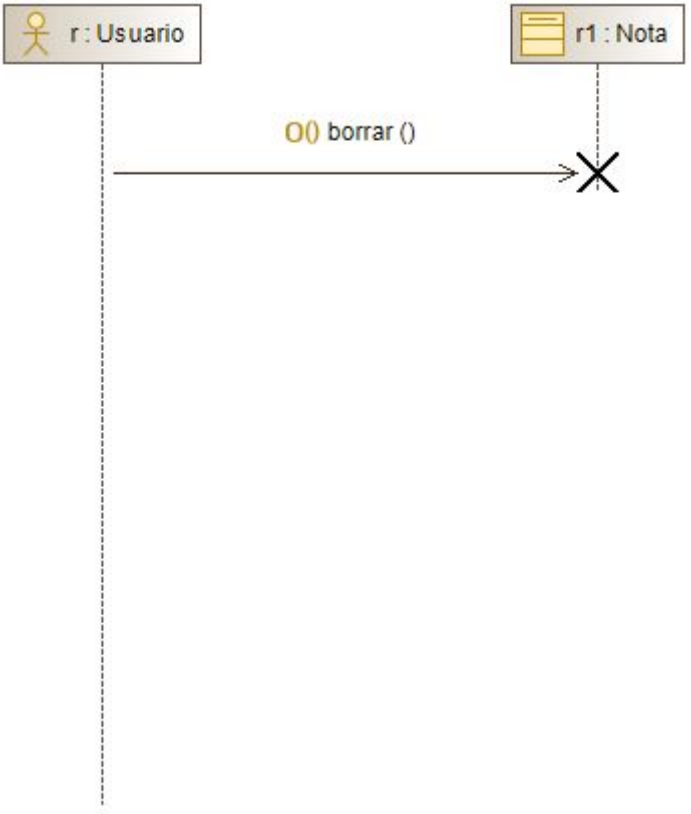


4.2-Diagramas de secuencia

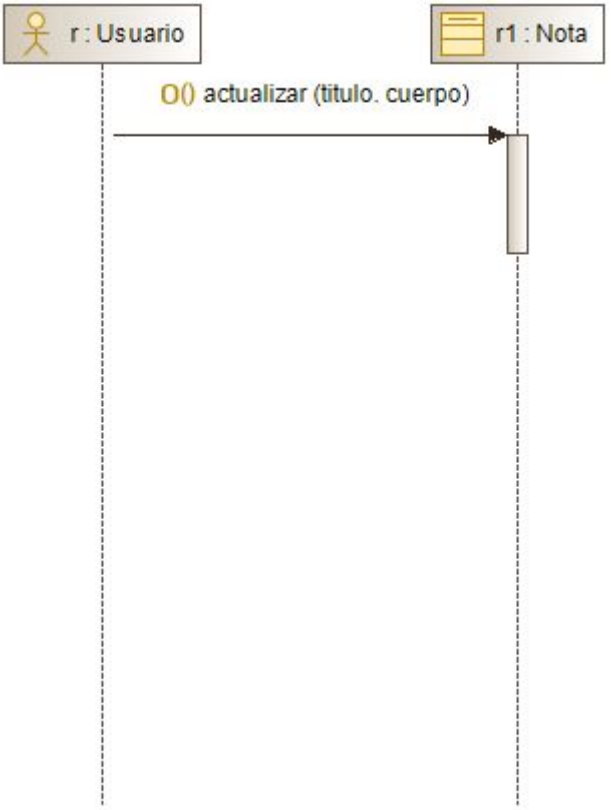
Crear nota



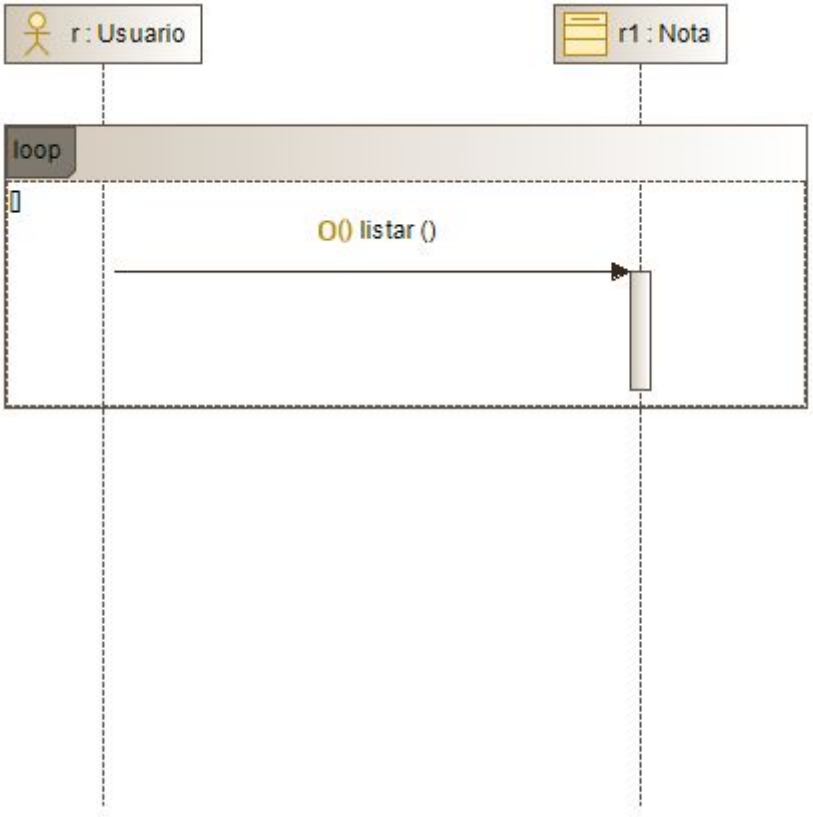
Borrar nota



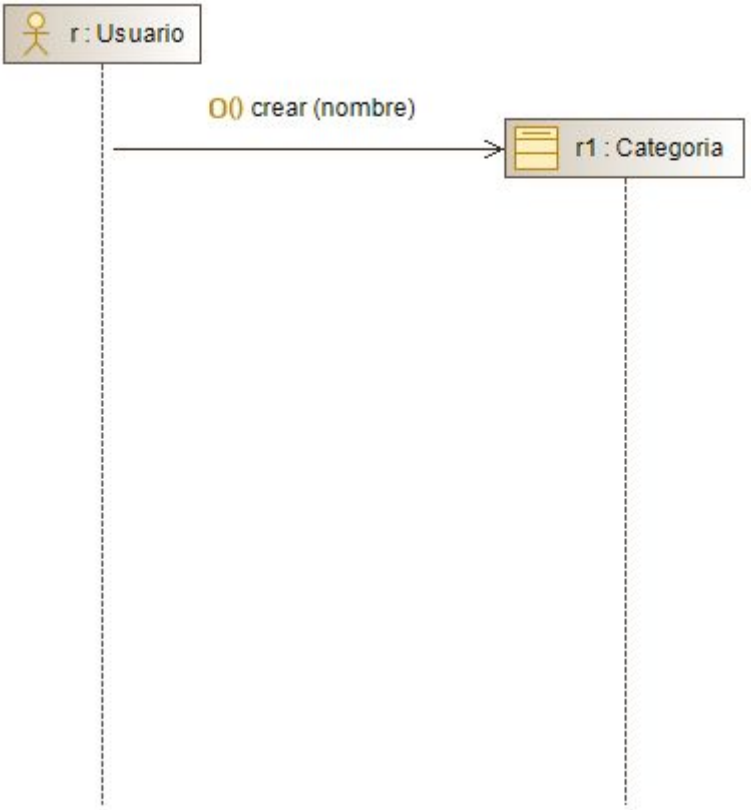
Actualizar nota



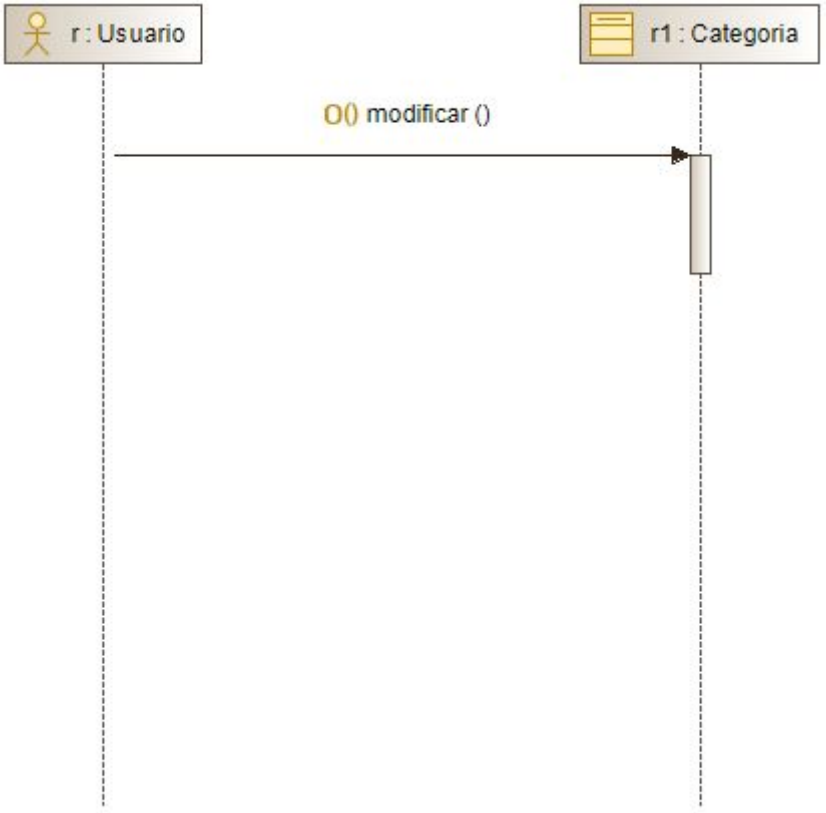
Consultar lista de notas ya existentes



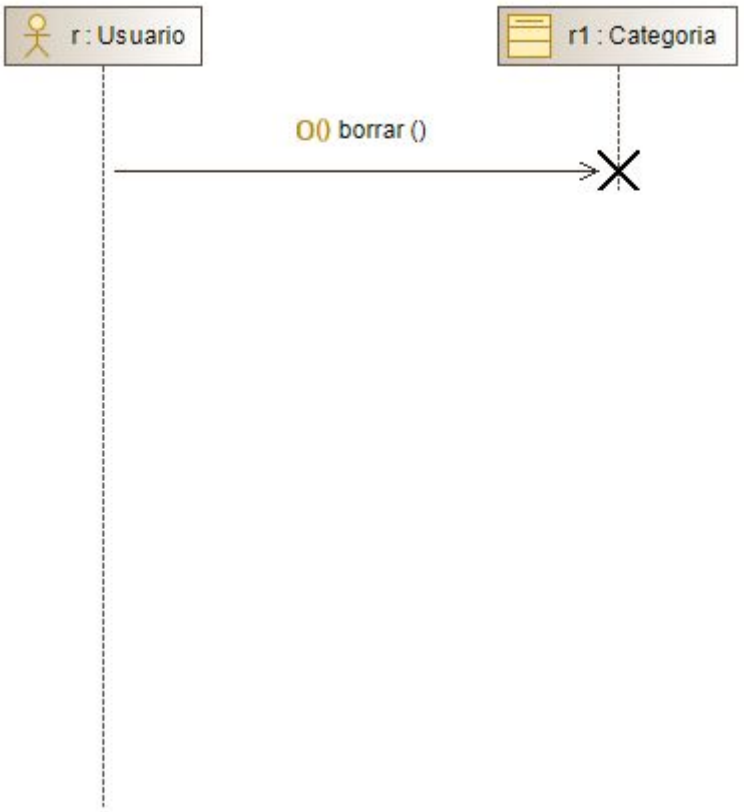
Crear categoría



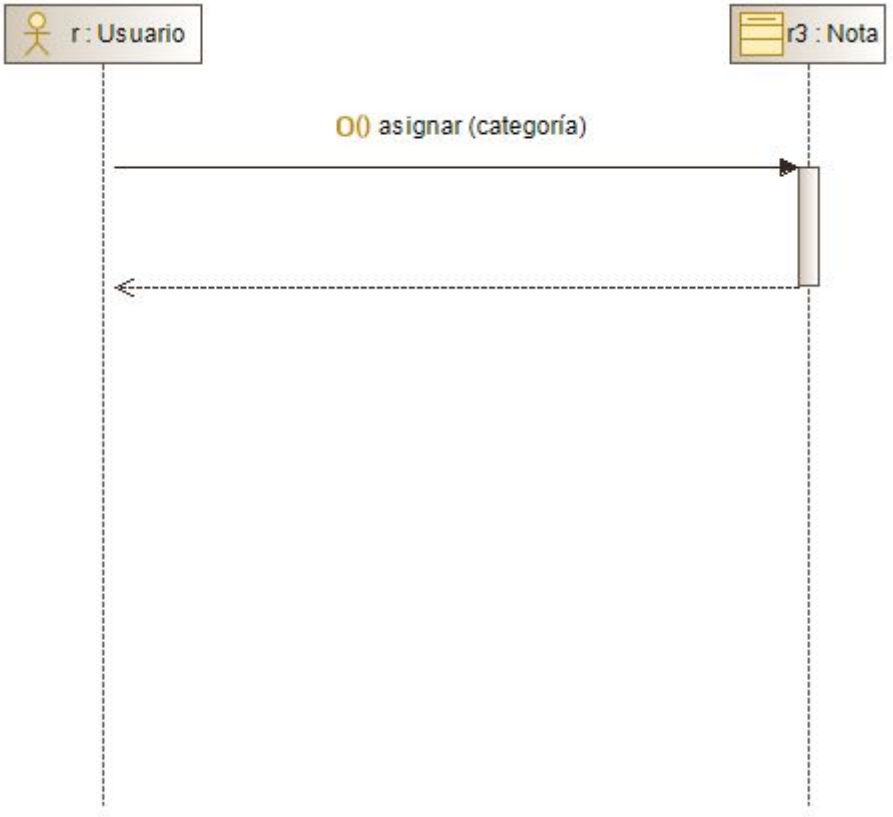
Modificar categoría



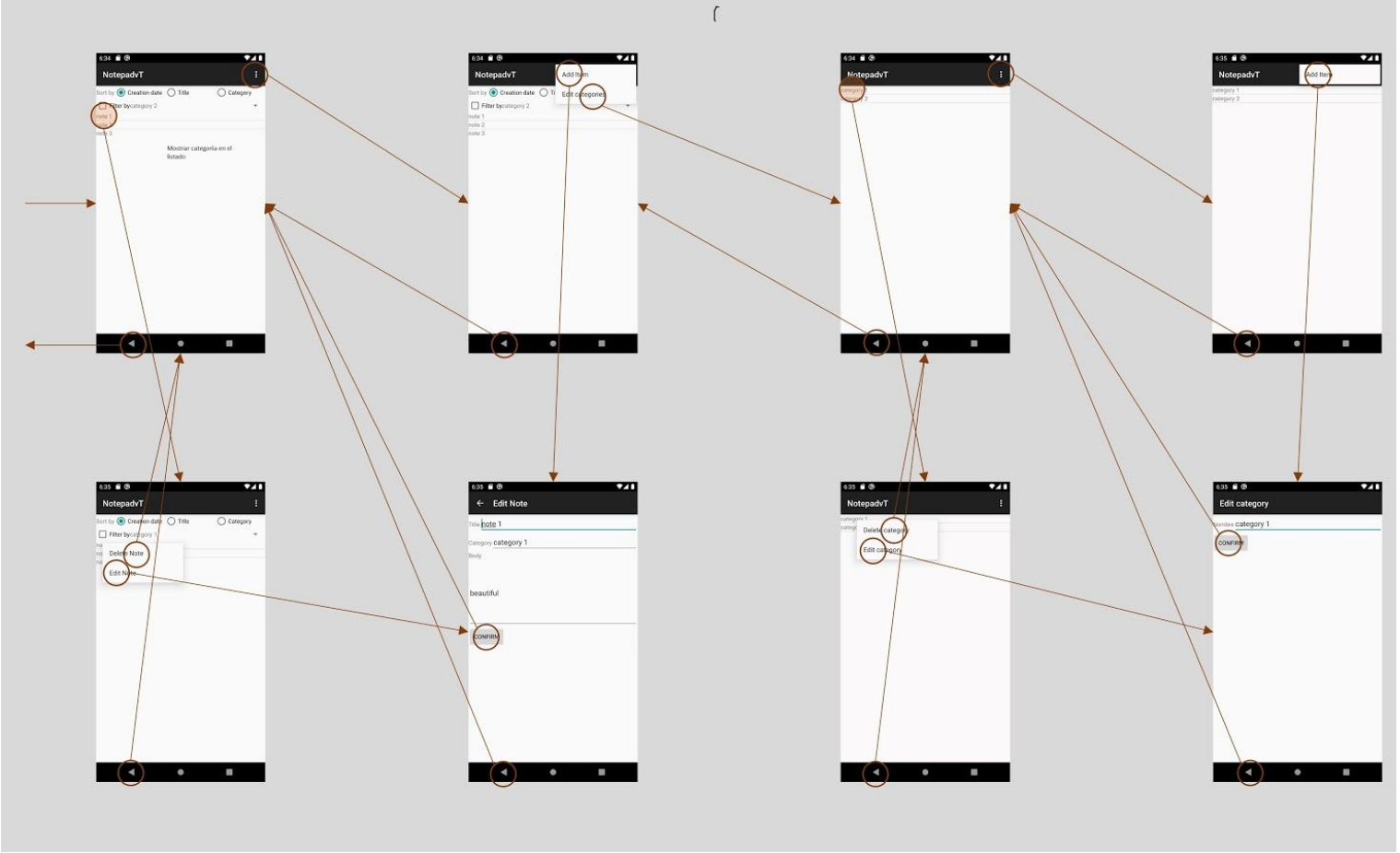
Eliminar categoría



Asignar categoría a nota ya existente

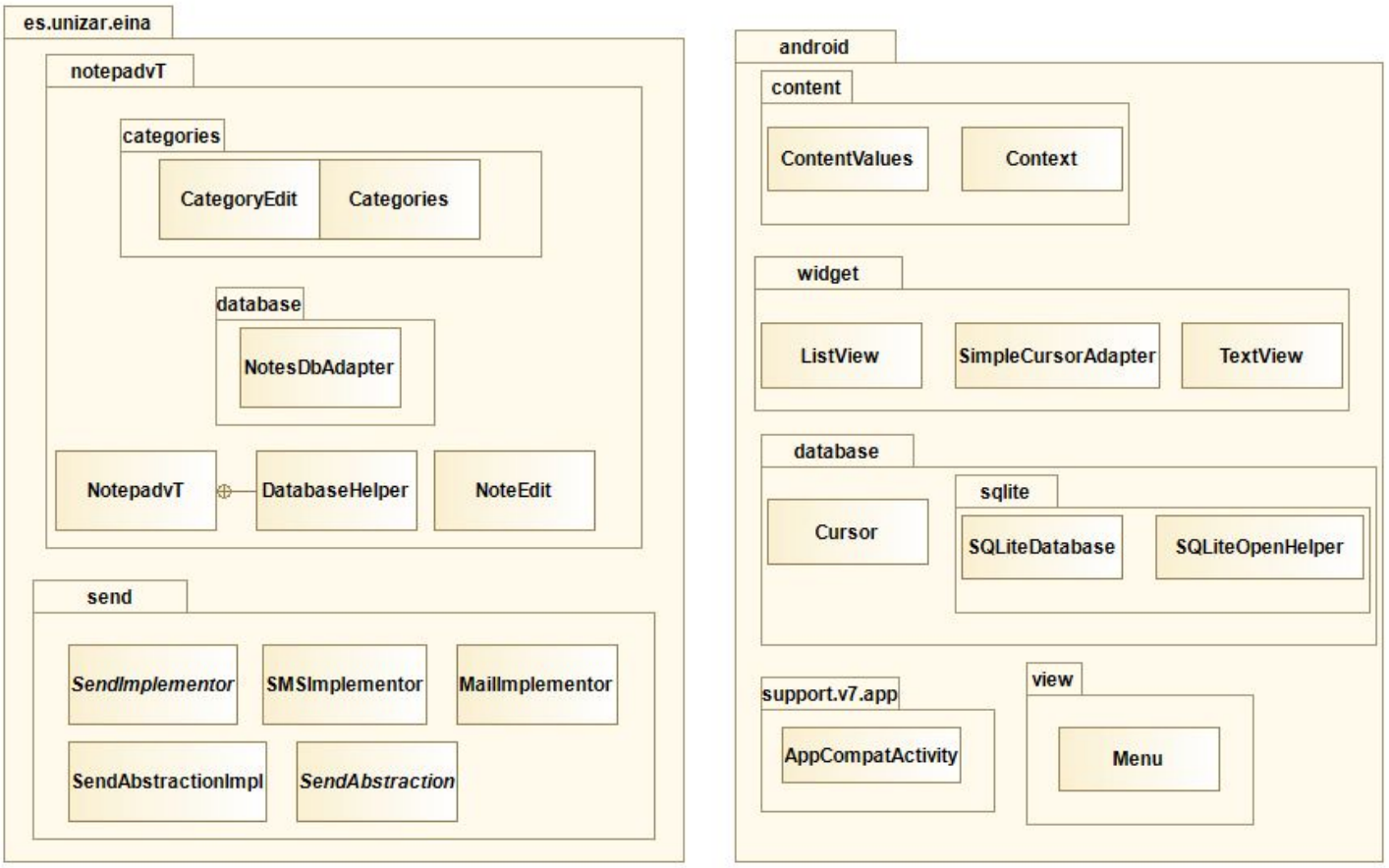


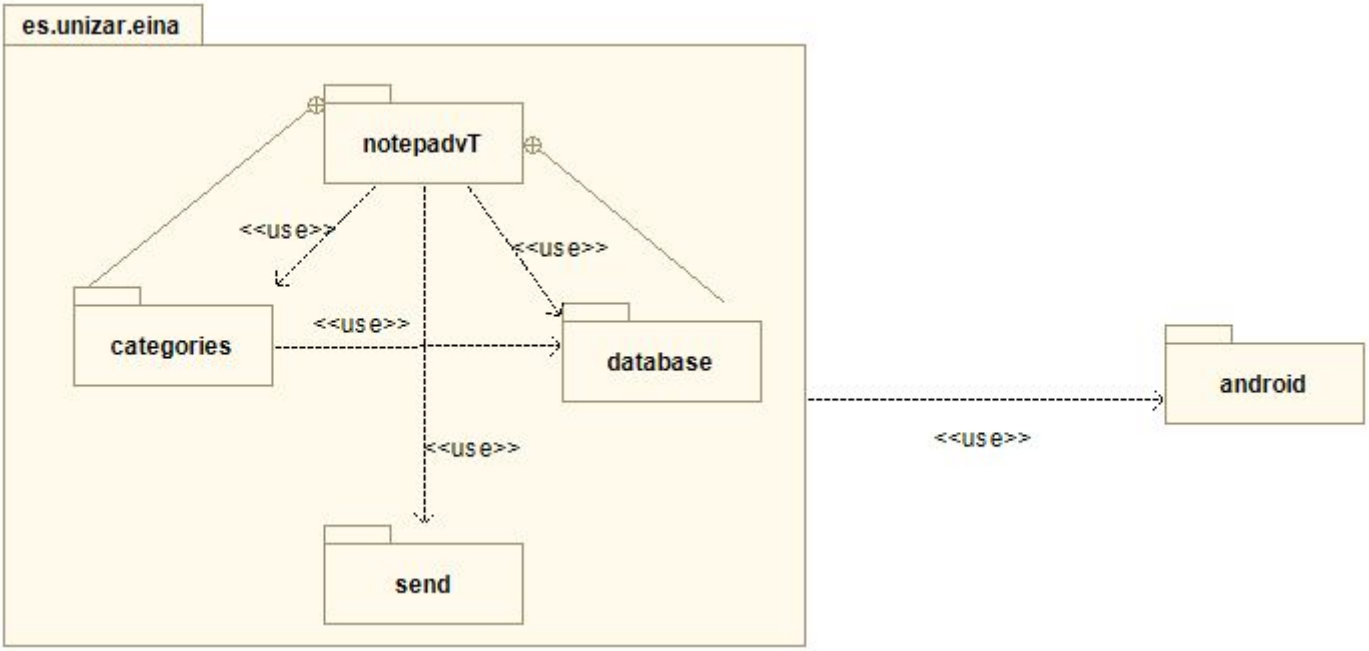
4.3-Mapa de navegación



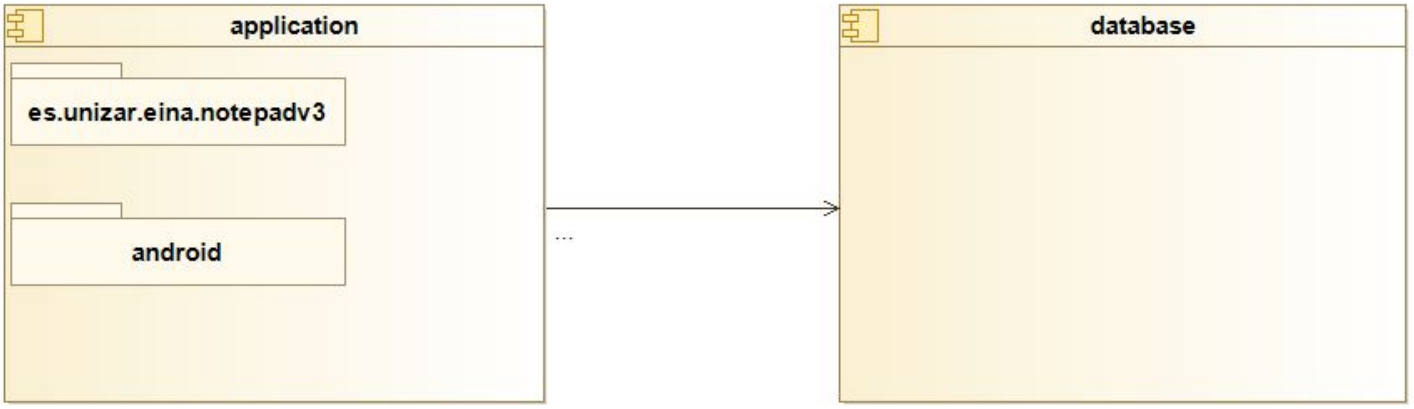
5.-Diagramas de fase de diseño del sistema

5.1-Diagrama de paquetes

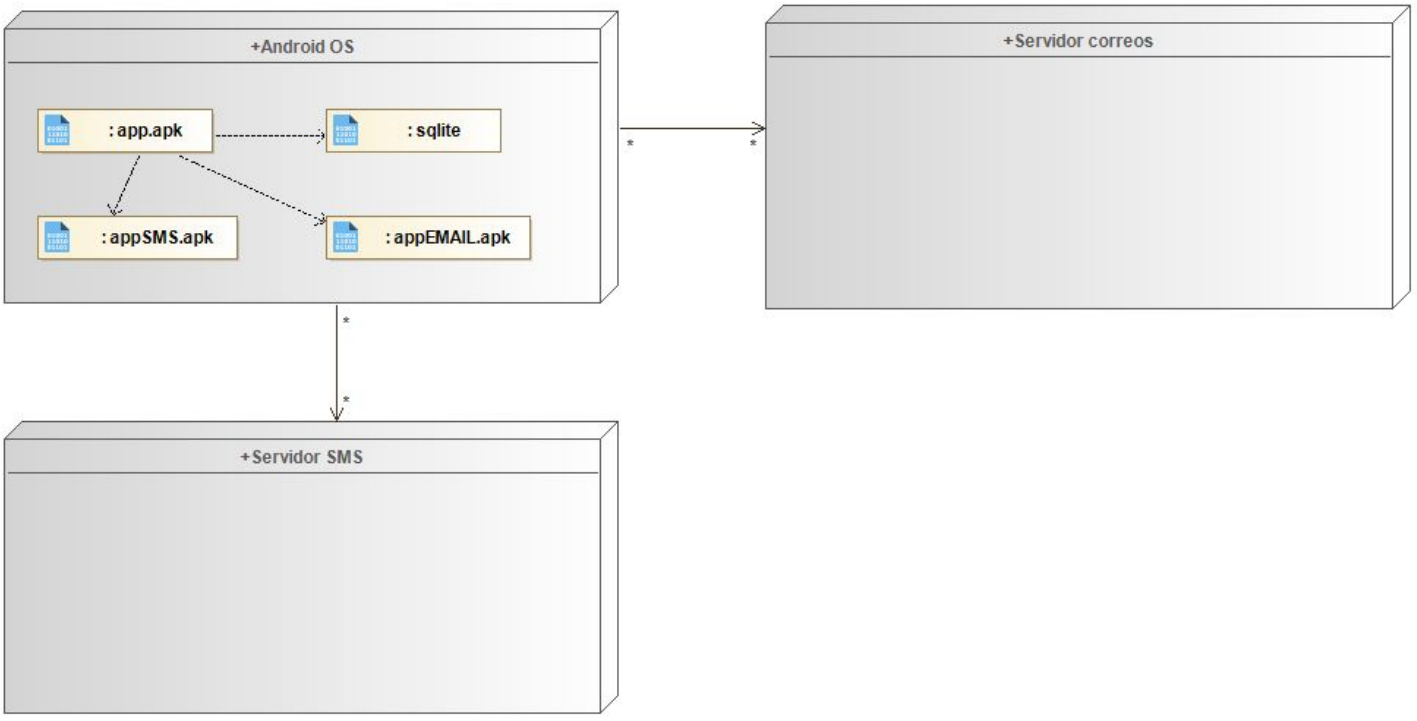




5.2-Diagrama de componentes



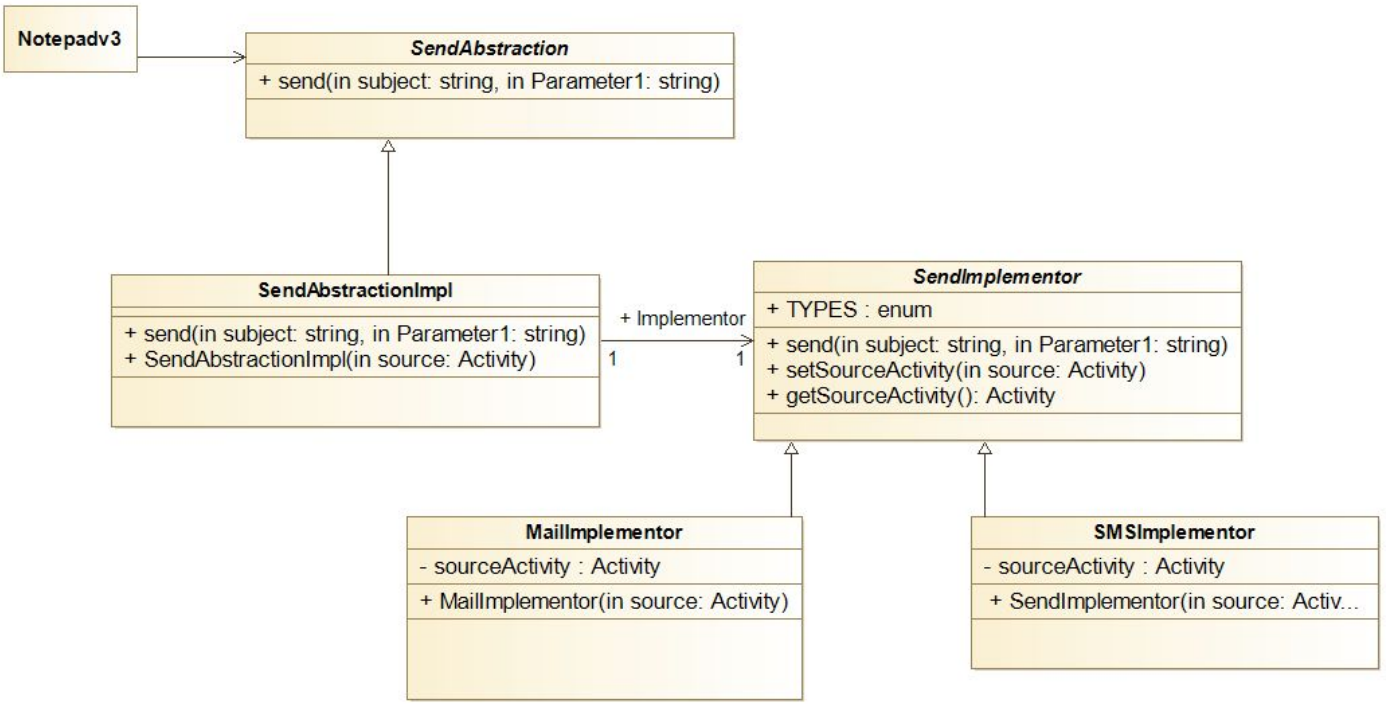
5.3-Diagrama de despliegue



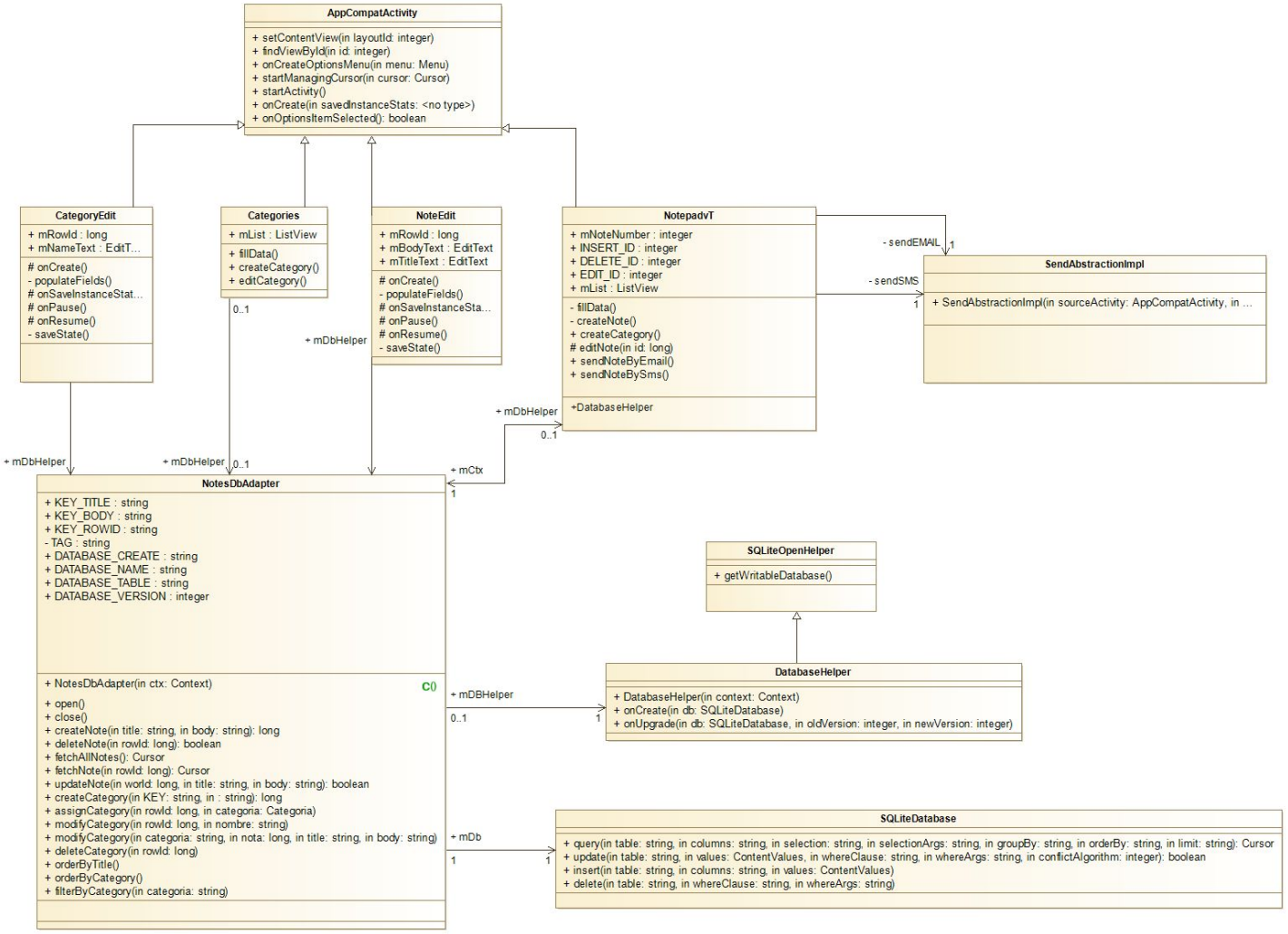
6.-Diagramas fase de diseño de objetos

6.1-Selección patrón de diseño

Siguiendo las recomendaciones del guión de prácticas se ha aplicado el patrón bridge, que nos permite separar las peticiones de SMS y EMAIL mediante una estructura de herencia y clases abstractas.

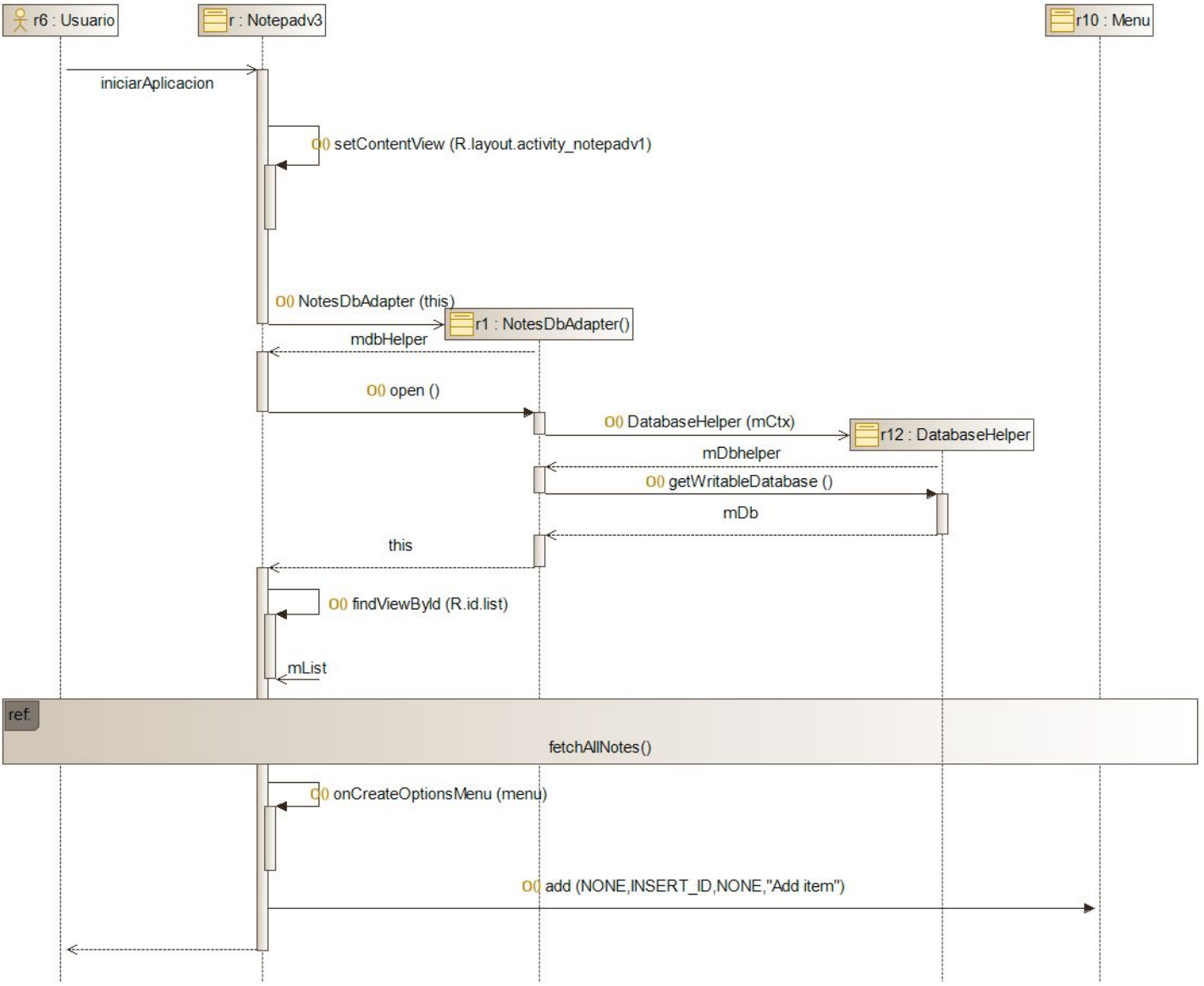


6.2-Diagrama de clases

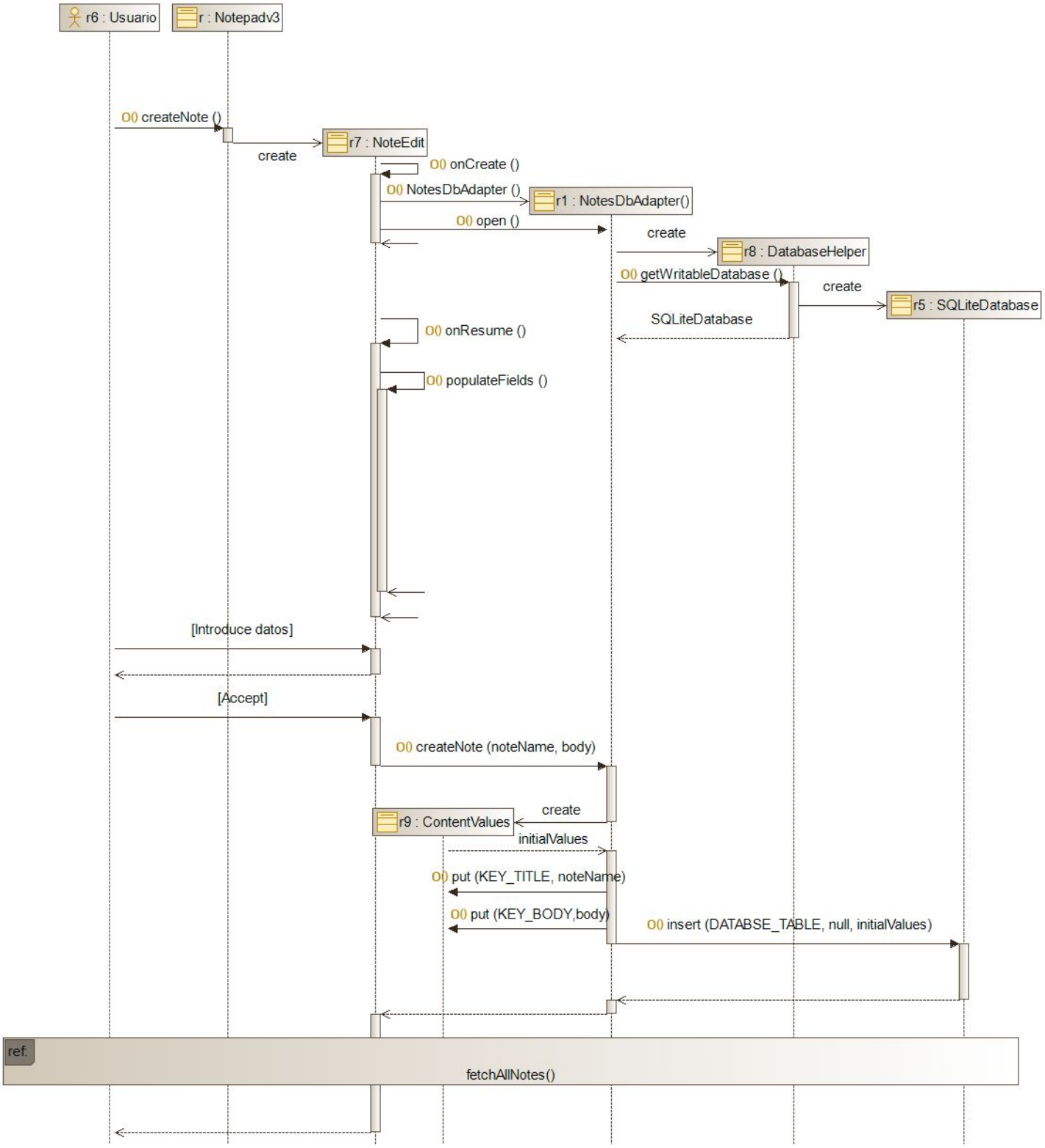


6.3-Diagramas de secuencia

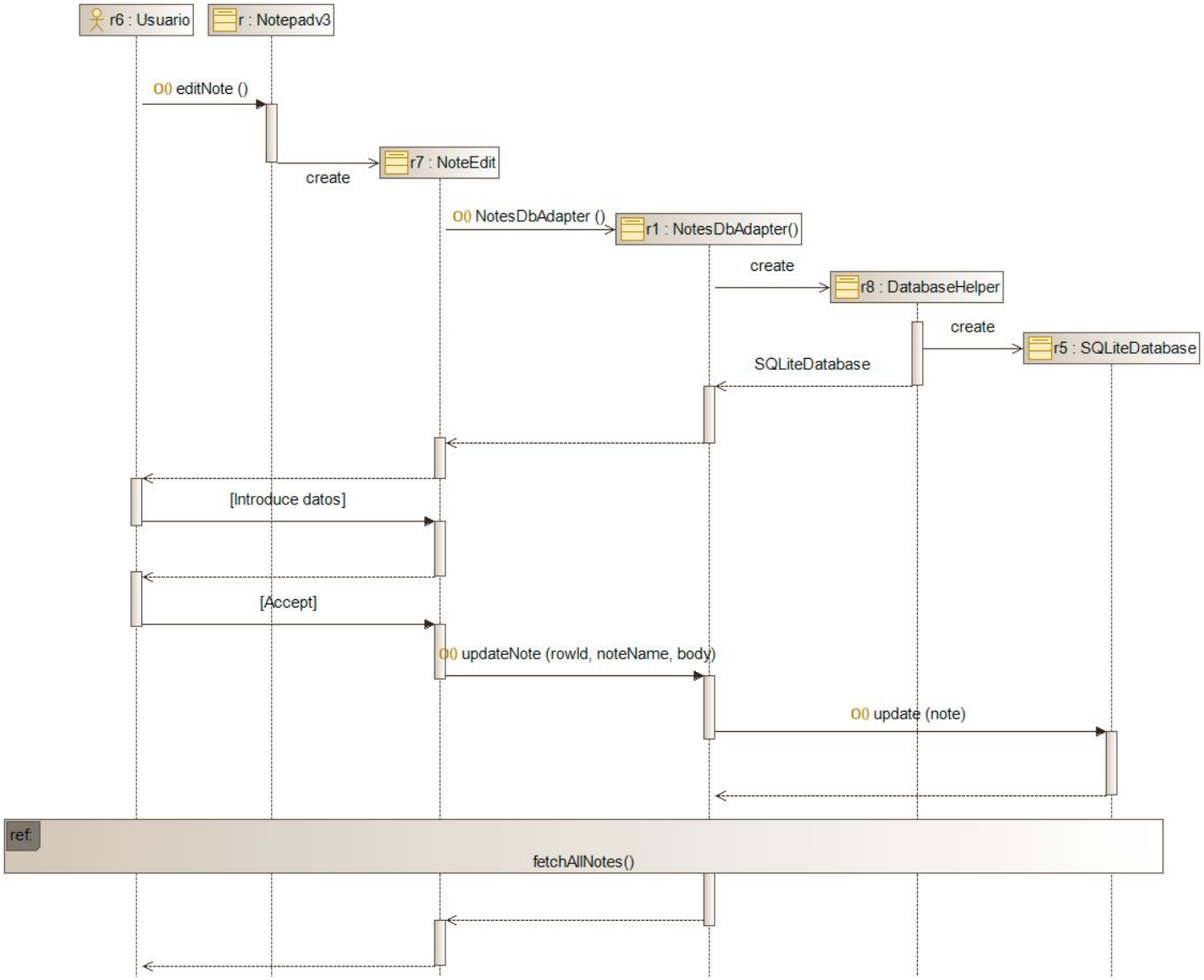
Inicialización de la aplicación



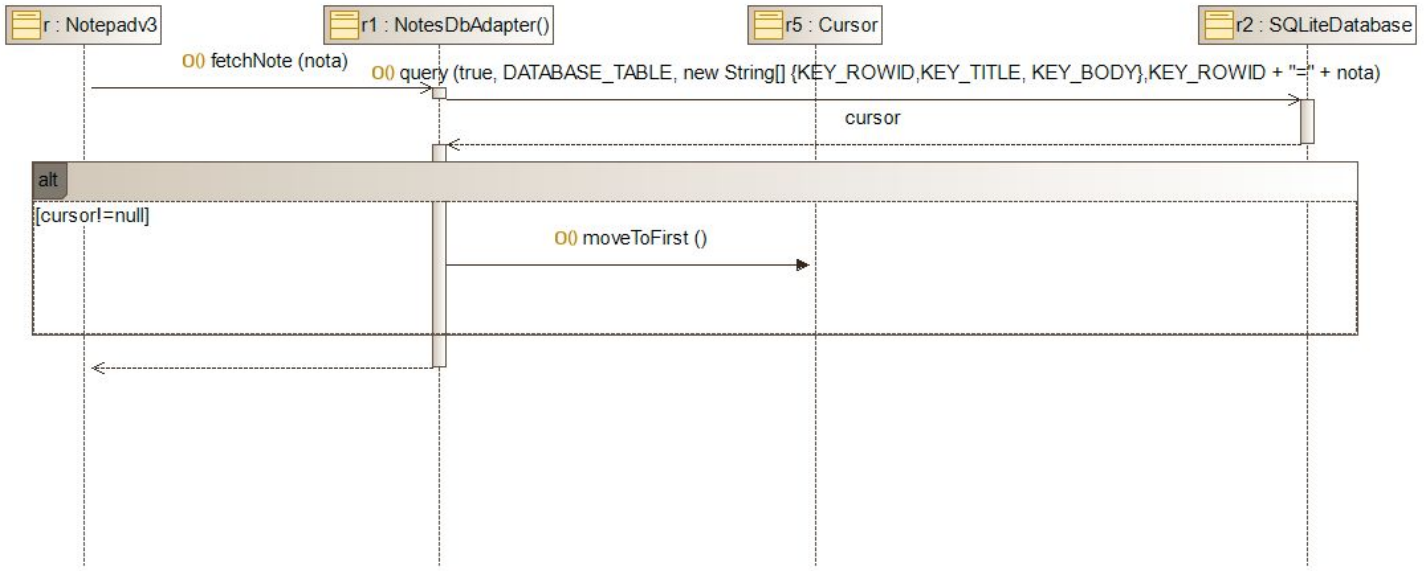
Crear nota


r6 : Usuario
r : Notepadv3
createNote ()
create
r7 : NoteEdit
onCreate ()
NotesDbAdapter ()
r1 : NotesDbAdapter()
open ()
create
r8 : DatabaseHelper
getWritableDatabase ()
create
r5 : SQLiteDatabase
SQLiteDatabase
onResume ()
populateFields ()
[Introduce datos]
[Accept]
createNote (noteName, body)
create
r9 : ContentValues
initialValues
put (KEY_TITLE, noteName)
put (KEY_BODY,body)
insert (DATABSE_TABLE, null, initialValues)
ref
fetchAllNotes()

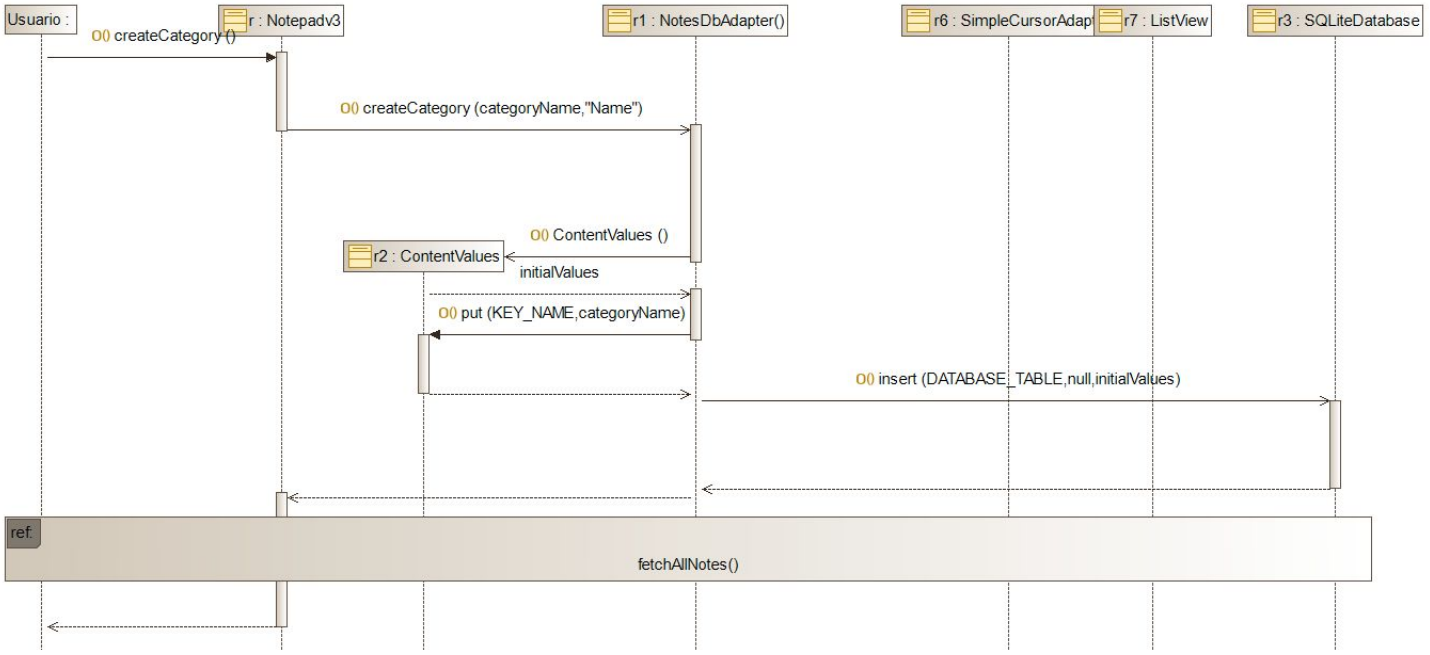
Actualizar nota



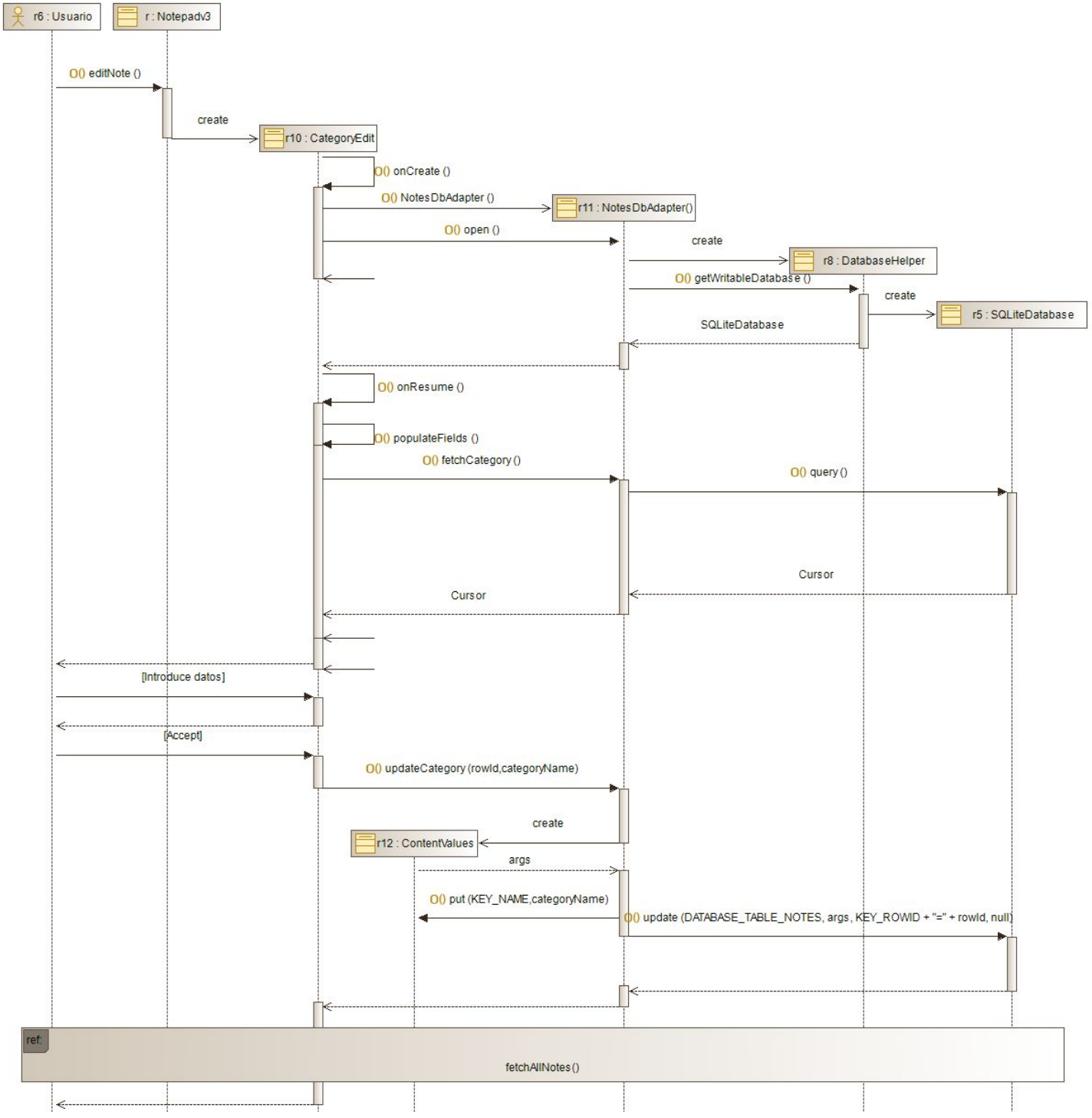
Consultar lista de notas ya existentes



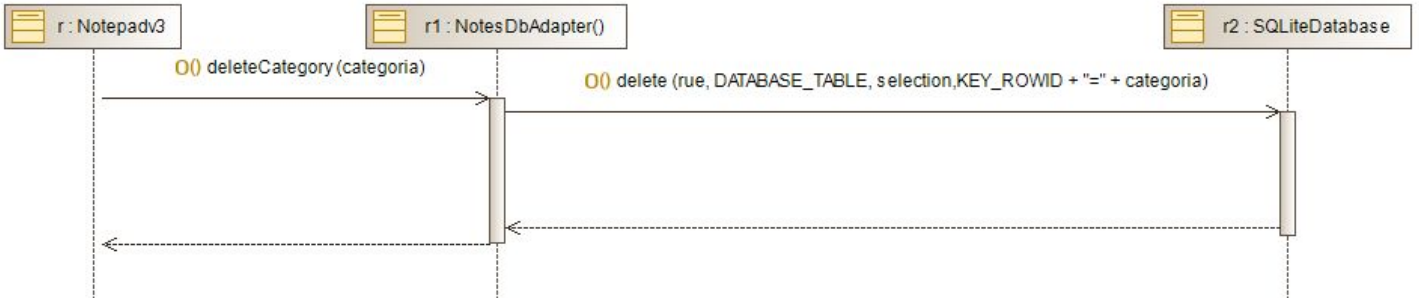
Crear categoría



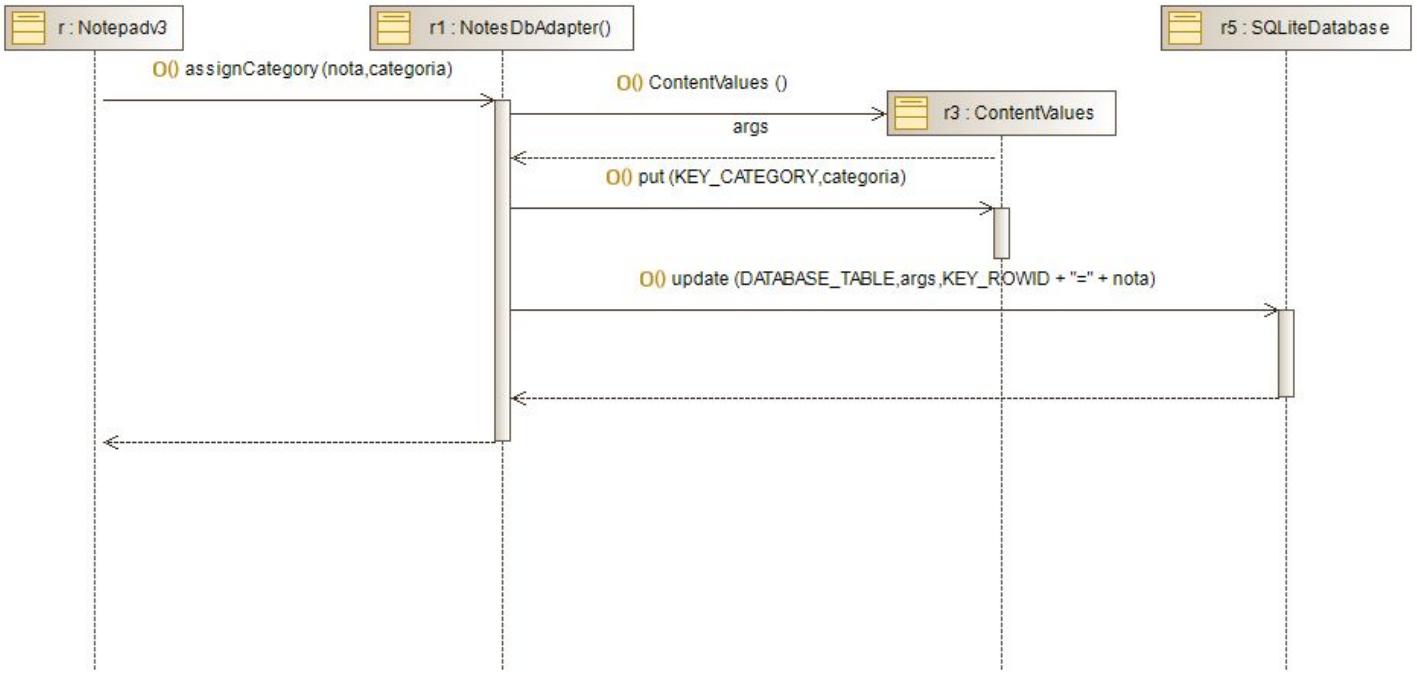
Modificar categoría



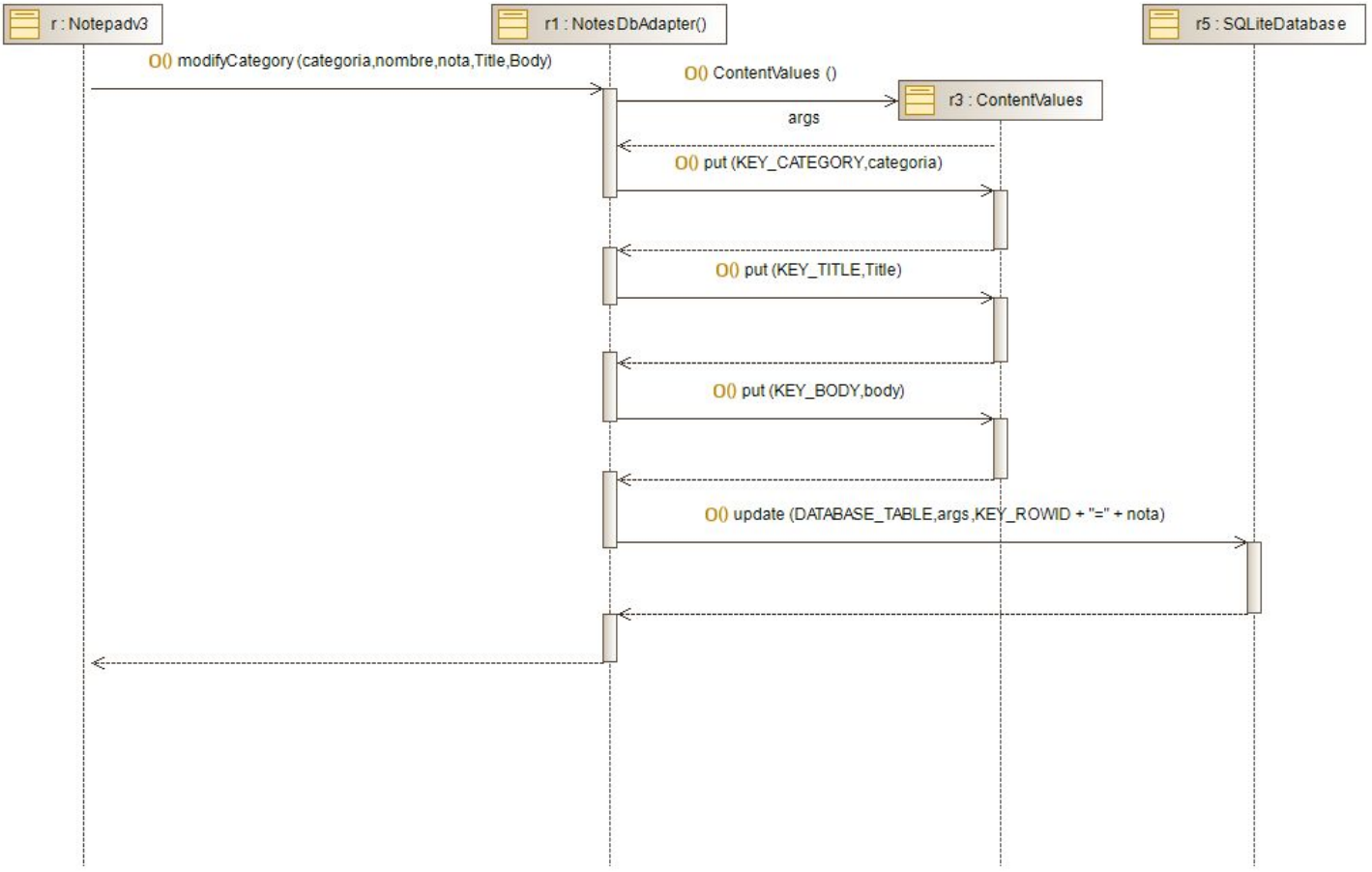
Eliminar categoría



Asignar categoría a nota ya existente

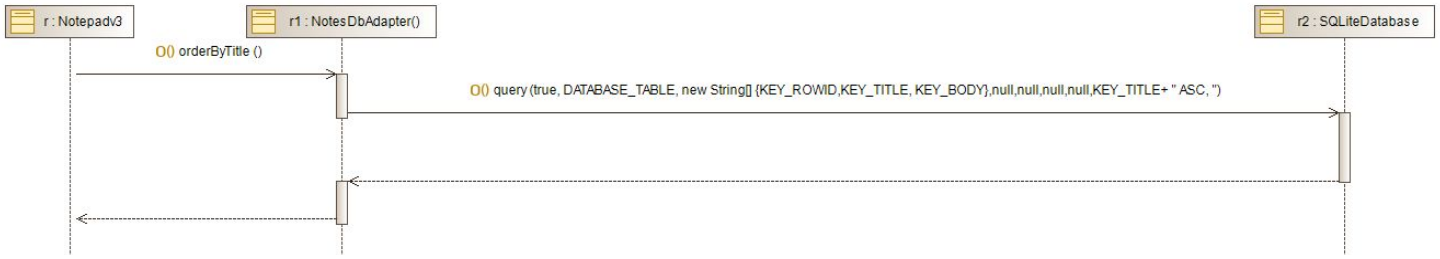


Modificar categoría asignada a nota existente



Ordenar lista de notas por categoría

Ordenar lista de notas por título



Filtrar lista de notas por categoría

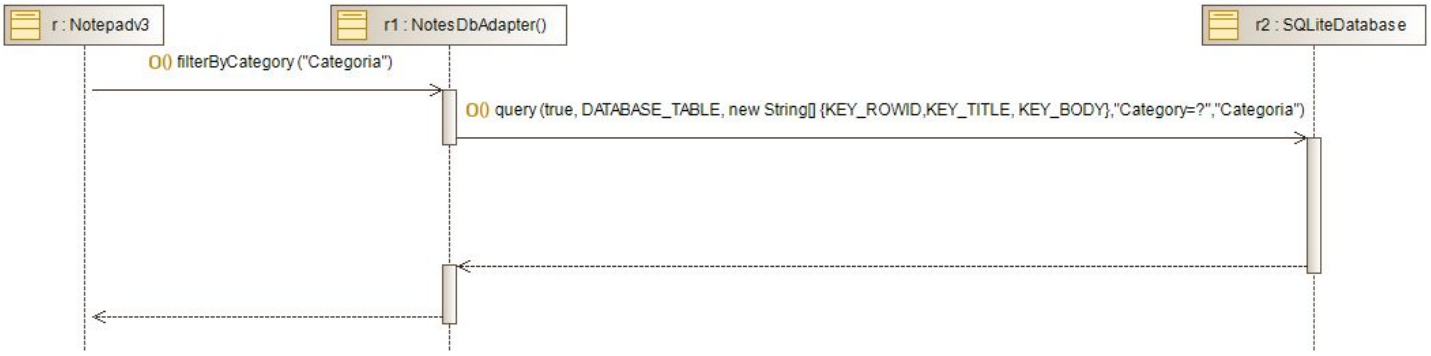
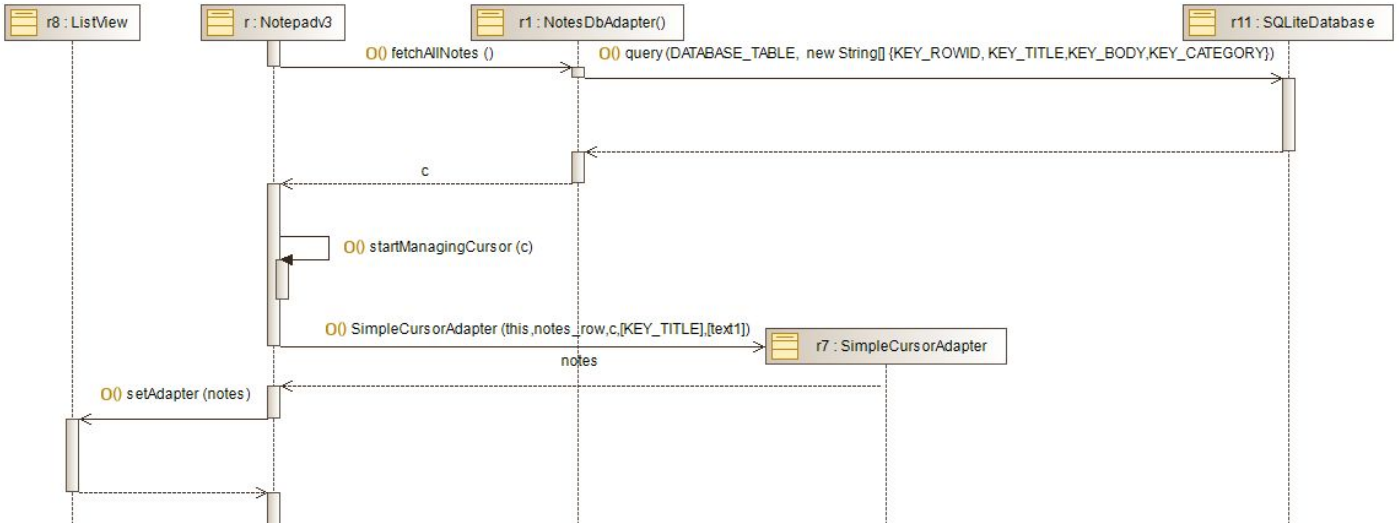
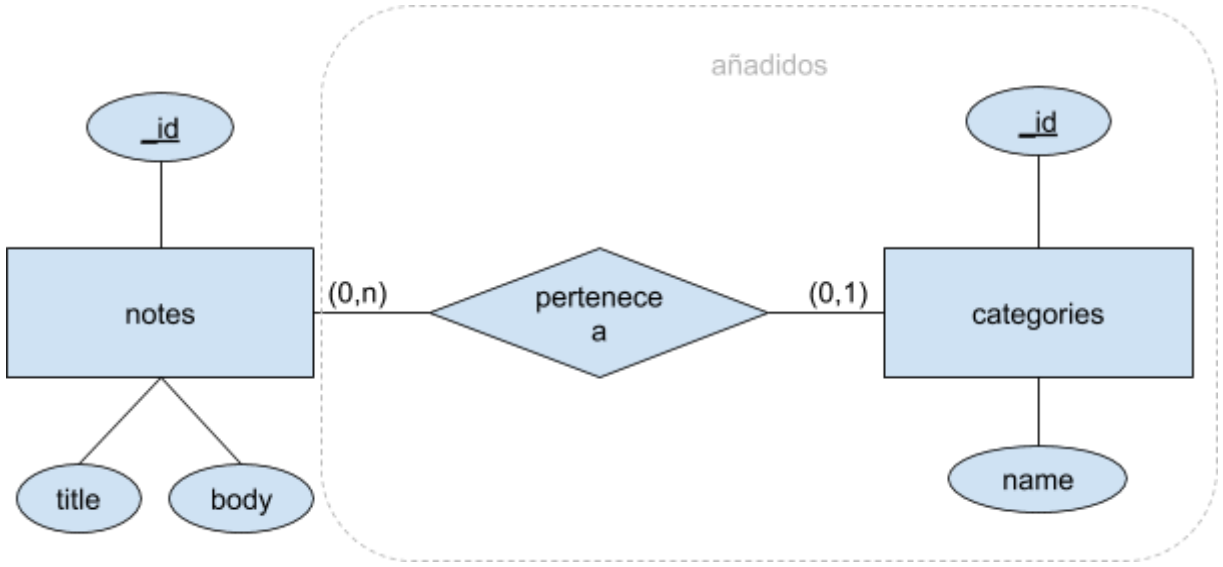


Diagrama de fetchAllNotes

Aparece referenciado en varios diagramas de secuencia.



6.4-Modelo entidad-relación



7.-Pruebas

7.1-Pruebas de caja negra

-Funcion public long createNote(String title, String body, Integer category)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
title	1: title.length()>0 2: title!=null	3: title.length()<=0 4: title==null
body	5: title!=null	6: title==null
category	7: category>=0 8: category=null	9: category<0

Entrada			Objetivo: clases cubiertas	Resultado
title	body	category		
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)				
"Hola"	"Hola"	1	1,2,5,7	rowId
"Hola"	"Hola"	null	1,2,5,8	rowId
Casos de prueba para clases de equivalencia no válidas (uno por clase)				
null	"Hola"	1	4	-1
"Hola"	null	1	6	-1
""	"Hola"	1	3	-1
"Hola"	"Hola"	-1	9	-1

-Funcion public boolean deleteNote(long rowId)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
rowId	1: rowId>=0	2: rowId<0

Entrada	Objetivo: clases cubiertas	Resultado
rowId		
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)		
2	1	true
Casos de prueba para clases de equivalencia no válidas (uno por clase)		
-1	2	false

-Funcion public boolean updateNote(long rowId, String title, String body, Integer category)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
title	1: title.length()>0 2: title!=null	3: title.length()<=0 4: title==null
body	5: title!=null	6: title==null
rowId	7: rowId>=0	8: rowId<0
category	9: category>=0 10: category=null	11: category<0

Entrada				Objetivo: clases cubiertas	Resultado
title	body	rowId	category		
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)					
"Hola"	"Hola"	1	1	1,2,5,7,9	true
"Hola"	"Hola"	1	null	1,2,5,7,10	true
Casos de prueba para clases de equivalencia no válidas (uno por clase)					
null	"Hola"	1	1	4	false
"Hola"	null	1	1	6	false
""	"Hola"	1	1	3	false
"Hola"	"Hola"	-1	1	3	false
"Hola"	"Hola"	1	-1	11	false

-Funcion public Cursor fetchAllNotes(String orderBy, String category)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
orderBy	1: orderBy=KEY_TITLE 2: orderBy=KEY_CATEGORY 3: orderBy=null	4: orderBy="ERROR"
category	5: category.length()>0 6: category.length()==0 7: category=null	

Entrada		Objetivo: clases cubiertas	Resultado
orderBy	Category		

Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)			
KEY_TITLE	"categoría"	1,5	Cursor
KEY_CATEGORY	""	2,6	Cursor
null	null	3,7	Cursor
Casos de prueba para clases de equivalencia no válidas (uno por clase)			
"ERROR"	null	4	null

-Funcion public Cursor fetchNote(long rowId)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
rowId	1: rowId>=0	2: rowId<0

Entrada rowId	Objetivo: clases cubiertas	Resultado
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)		
1	1	Cursor
Casos de prueba para clases de equivalencia no válidas (uno por clase)		
-1	2	null

-Funcion public long createCategory(String name)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Name	1: name.length()>0 2: name!=null	3: name.length()<=0 4: name==null

Entrada Name	Objetivo: clases cubiertas	Resultado
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)		
"Hola"	1,2	rowId
Casos de prueba para clases de equivalencia no válidas (uno por clase)		
null	4	-1
""	3	-1

-Funcion public boolean deleteCategory(long rowId)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
rowId	1: rowId>=0	2: rowId<0

Entrada Cuerpo	Objetivo: clases cubiertas	Resultado
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)		
2	1	true
Casos de prueba para clases de equivalencia no válidas (uno por clase)		
-1	2	false

-Funcion public Cursor fetchCategory(long rowId)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
rowId	1: rowId>=0	2: rowId<0

Entrada rowId	Objetivo: clases cubiertas	Resultado
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)		
1	1	Cursor
Casos de prueba para clases de equivalencia no válidas (uno por clase)		
-1	2	null

-Funcion public Integer fetchCategory(string name)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Name	1: name.length()>0 2: name!=null	3: name.length()<=0 4: name==null

Entrada Name	Objetivo: clases cubiertas	Resultado
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)		
"Hola"	1,2	rowId
Casos de prueba para clases de equivalencia no válidas (uno por clase)		

null	4	null
“”	3	null

-Funcion public boolean updateCategory(long rowId, String name)

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
Name	1: title.length()>0 2: title!=null	3: title.length()<=0 4: title==null
rowId	5: rowId>=0	6: rowId<0

Entrada		Objetivo: clases cubiertas	Resultado
Name	rowId		
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)			
“Hola”	1	1,2,5	true

Casos de prueba para clases de equivalencia no válidas (uno por clase)			
null	1	4	false
“Hola”	-1	6	false
“”	1	3	false

7.2-Prueba de volumen

Para las pruebas de volumen añadimos una requisito no funcional, “El sistema no puede almacenar más de 1000 notas”, he intentamos comprobar este requisito mediante otro test como los realizados en la prueba de caja negra del apartado anterior.

Entrada	Clases de equivalencia válidas	Clases de equivalencia no válidas
n (n° notas a crear)	1: n<1000	2: n>=1000

Entrada	Objetivo: clases cubiertas	Resultado
n		
Casos de prueba para clases de equivalencia válidas (n° mínimo de casos de prueba)		
500	1	operacion realizada

Casos de prueba para clases de equivalencia no válidas (uno por clase)		
1000	2	operacion realizada

Como se puede ver en la prueba de caja negra de arriba, nuestra aplicación puede crear más de 1000 notas, no está limitado por ningún número más que por la capacidad del dispositivo.

7.3- Análisis de los resultados

¿Funcionan los casos de prueba correspondientes a particiones de equivalencia válidas?

Los casos de prueba de las particiones de equivalencias válidas sí funcionan, para los casos de pruebas que cubren las clases válidas devuelven el resultado esperado, todos pasan los tests sin problemas.

¿Y los de particiones de equivalencia no válidas?

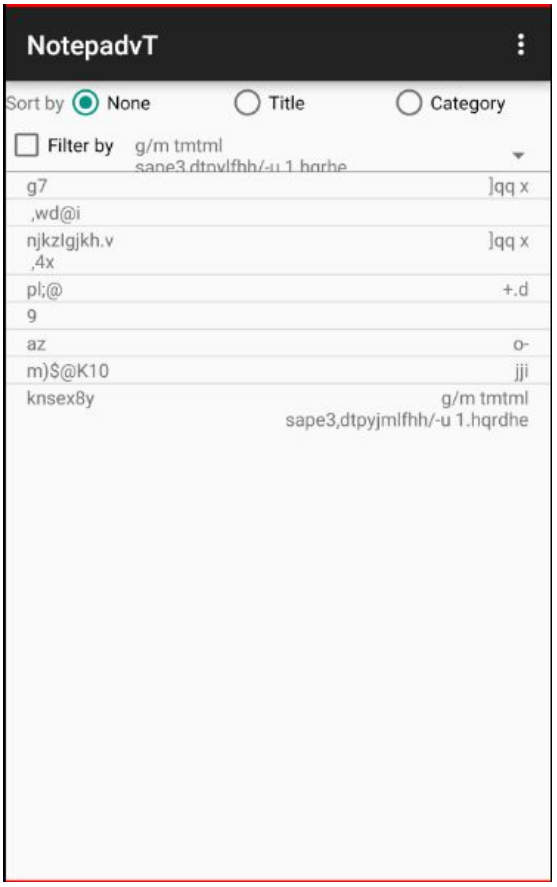
Los casos de prueba de las particiones de equivalencia no válidas fallan en algunos casos (cuando el campo title y body son null o cuando el title es la cadena vacía “”). Se ha modificado el código para corregir estos fallos.

¿Se ha producido algún error? ¿Ha aparecido algún problema de usabilidad?

Tras la primera ejecución de los test fallaban aquellos de las clases de equivalencia no válidas, que se corrigieron posteriormente. Tras la ejecución de los test de usabilidad y por indicación del enunciado se ha detectado que desplazarse en la lista de notas es tedioso, por lo que se ha implementado la característica que hace que la lista se autodesplace a la nota editada/creada/borrada. Todos los errores se han corregido y las pruebas son satisfactorias.

7.4-Pruebas de sobrecarga (monkey)

Resultados de aplicar monkey con 50000 eventos `adb shell monkey -p es.unizar.eina.notepadvT -v -s 1573333539477 50000`



Monkey finaliza sin errores y devuelve por pantalla:

Events injected: 50000
:Sending rotation degree=0, persist=false
:Dropped: keys=43 pointers=36 trackballs=0 flips=0 rotations=0
Network stats: elapsed time=228894ms (0ms mobile, 0ms wifi, 228894ms not connected)
// Monkey finished

Por lo tanto podemos suponer que nuestra aplicación pasa la prueba de sobrecarga sin problemas, ya que con 50000 eventos realizados por monkey no sucede ningún error ni se cierra la aplicación, como podemos observar en las imágenes toda categoría y nota es creada correctamente.

7.5- Descripción pruebas

Se han desarrollado dos tipos de pruebas (tres contando la ejecución de monkey).

1) Pruebas de base de datos: realizadas utilizando el framework junit disponible de forma nativa en android studio, accediendo directamente a la clase DBAdapter sin necesidad de iniciar la aplicación (junit genera un context equivalente, pero la interfaz no se llega a mostrar). Con esto se han probado los tests de caja negra. El código de aplicación no ha sido modificado para la realización de las pruebas, y por ello también se han incluido utilidades para test manuales que permiten crear muchas notas y borrar todas las notas.
2) Pruebas de interfaz: realizadas utilizando el framework expresso disponible de forma nativa en android studio. Estas pruebas han simulado los casos de uso de creación/borrado/modificación de notas, para comprobar que visualmente la aplicación se comporta adecuadamente.

8.-Resultados y conclusiones.

Gracias a este trabajo se ha podido comprobar de primera mano lo horrorosa que es la aplicación Modelio, y lo inútil que es realizar diagramas de análisis y diseño no-personales. El uso de diagramas es recomendable, y en ocasiones obligatorio, como explicación o documentación sobre algo ya existente. También es útil realizar esquemas en papel para uso personal o como método de comunicación entre miembros de un equipo, pero en ningún caso los diagramas deben ser la finalidad del proyecto, como aquí se ha obligado.
Por contra se ha podido comprobar la sencillez y cantidad de atajos y facilidades que Android Studio proporciona, así como lo agradable que es programar en dicho ecosistema.
Como valoración personal este trabajo debería dejarse realizar en otros programas que no sean Modelio, y centrarse menos en la parte de diagramas que, si bien es importante, su uso en ambientes reales (no académicos) no es ni de cerca el que se enseña.

9.-Bibliografía

https://www.modelio.org/quick-start-pages-35.html

https://stackoverflow.com/

https://developer.android.com/studio/intro

https://github.com/