Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Diseño del Sistema

Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android

Jimena Patricia Alcántara Olivares
Matricula: 207333058

José Daniel López Jaimes
Matricula: 207333480

Asesor: Dra. María Lizbeth Gallardo López
Profesor Investigador
Departamento de Sistemas

Enero 2013

Índice

1 Visión

1.1 Historia de Revisiones

Versión del Documento	Fecha	Descripción de la Versión	Autores
1.0	09/02/2012	Inicio proyecto	Jimena P. Alcántara Olivares José Daniel López Jaimes

Versión del Documento	Fecha	Descripción de la Versión	Autores
2.0	07/03/2012	Ampliación de los objetivos de alto nivel	Jimena P. Alcántara Olivares José Daniel López Jaimes

Versión del Documento	Fecha	Descripción de la Versión	Autores
3.0	09/06/2012	Cambios en las consideraciones del sistema y en objetivos de alto nivel.	Jimena P. Alcántara Olivares José Daniel López Jaimes

1.2 Introducción

Este proyecto se enfoca a la realización de una **aplicación colaborativa** llamada *SHAREDRAW* que facilitará la interacción entre los miembros de un grupo de trabajo en la edición de figuras geométricas, así como en la edición texto. La aplicación será de tipo ***groupware*** y podrá ser ejecutada en dispositivos móviles con Sistema Operativo (S.O.) *Android.*

En este documento existen distintas secciones que se describirán a continuación:

i. Historia de revisiones: Se tiene una tabla para apuntar todas las revisiones que se lleven a cabo, describiendo los datos más importantes que se lleven a cabo en dicha versión.
ii. Introducción: Da el propósito del proyecto y descripción del documento de visión.
iii. Orientación: Se describe el giro del negocio y su principal necesidad (problema), así como forma de operar del negocio.
iv. Descripción del personal involucrado: Se describe en esta zona el personal: parte del cliente, parte del equipo de desarrollo; los objetivos de alto nivel y soluciones actuales por parte del cliente.
v. Visión general del sistema computacional: perspectiva del sistema, así como sus beneficios.
vi. Resumen de las características del sistema: requisitos funcionales del sistema.

1.3 Orientación

1.3.1 Antecedentes

En la actualidad muchos proyectos tienen la necesidad de que los miembros de un equipo trabajen al mismo tiempo sobre alguna aplicación, esto implica desarrollar aplicaciones colaborativas donde los miembros puedan compartir y editar el avance general. Tomando en cuenta que no siempre se tiene acceso a una computadora personal y que la evolución de los dispositivos móviles permite realizar **"tareas robustas"** es de interés utilizar dichos dispositivos como una herramienta para trabajar a cierta distancia con las demás personas.

1.3.2 Enunciado del Problema

Los usuarios tienen la necesidad de trabajar de manera grupal sobre algún proyecto en tiempo real, mediante distintas computadoras.

En la actualidad se ha exaltado la importancia de migrar aplicaciones de terminales a dispositivos móviles para aportar una mayor portabilidad al usuario.

1.3.3 Enunciado de Posición

La aplicación estará dirigida principalmente a usuarios que cuenten con dispositivos móviles con sistema operativo Android desde la versión 2 en adelante.

1.4 Descripción del personal involucrado

1.4.1 Personal por parte del desarrollo de la Aplicación

Alcántara Olivares Jimena Patricia.

López Jaimes José Daniel

1.4.2 Objetivos de Alto Nivel

Objetivo	Prioridad	Solución Actual	Problemas
Implementar aplicación SHAREDRAW para dispositivos móviles	Alta	Existe una aplicación solo para Computadora.	Falta de portabilidad
Gestión de figuras y texto	Alta	Aplicación para PC	Distintas clases, bibliotecas y enfoque de programación de Java a una programación directa en Android
Diseño de la aplicación	Alta	Aplicación para PC	La aplicación para PC

teniendo en cuenta las posibilidades y limitantes del S.O. Android; así como el manejo de una aplicación en este sistema.			no es atractiva visualmente, elemento que afecta directamente en la temática de la aplicación.
Realizar una comunicación entre dispositivos móviles bajo una red.	Alta	Existe una red cliente-servidor para computadora.	Falta de portabilidad a nivel de equipos que contienen la aplicación.
Diseño e implementación de un módulo de transición entre espacios.	Alta	Aplicación PC	La aplicación debe manejar las posibles excepciones y errores para satisfacer al usuario
Elaboración de una barra de herramientas para el manejo de la edición y comportamiento de los objetos.	Alta	Aplicación PC	Android brinda la posibilidad de crear barras de herramientas más atractivas en base a un diseño propio de íconos y botones(formas y tamaños).
Elaboración de un tutorial de ayuda dentro de la aplicación para explicar el funcionamiento de la misma	Baja	No hay solución	Dado que Android ofrece métodos de interfaz touch-screen[1] será útil un tutorial para guiar al usuario.

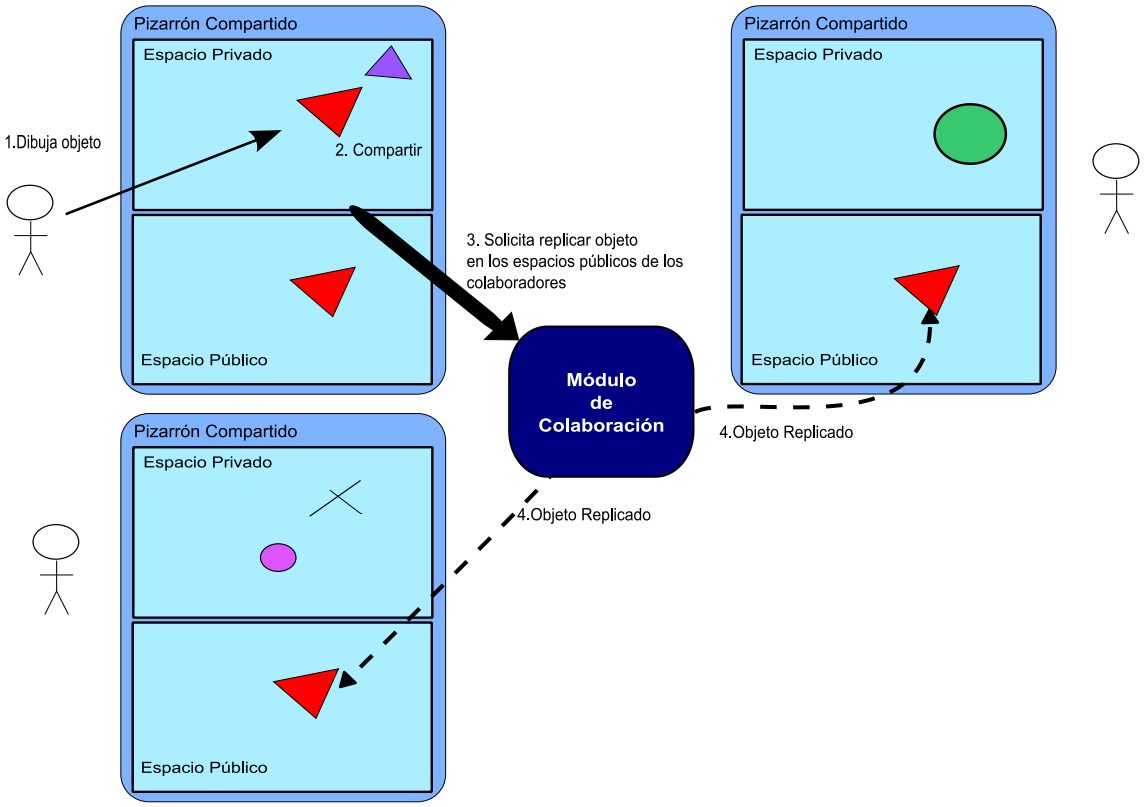
1.5 Objetivos a Nivel de Usuario

Los usuarios contarán con una aplicación que sea portable y con la cual podrán colaborar en la edición de figuras y texto como forma de entretenimiento dentro de una misma red.

[1]

1.6 Visión General del Sistema Computacional

1.6.1 ShareDraw



1.7 Resumen de las Características del Sistema

- Debido a la variedad de tamaños de pantalla que tienen los dispositivos actuales, la aplicación se encargará de mostrar una **"vista gráfica"** adecuada para el tamaño y condiciones de la pantalla del dispositivo móvil que sea utilizado.
- La aplicación incluirá las diversas opciones que podrá realizar el usuario enfocadas al objeto u objetos manipulados en la aplicación ShareDraw. Las opciones a elegir serán: *Crear* objetos, *Modificar* características de las figuras geométricas como tamaño y color; el tamaño y mensaje del objeto texto, *Borrar* objetos, *Duplicar* objetos.

- Para la manipulación de los objetos será necesario manejar la técnica de *multitouch*[2].
- La aplicación dará la opción de compartir sus objetos del usuario con los demás usuarios e importar los objetos al Espacio Privado.

La aplicación permitirá conectarse con otros usuarios mediante una red para colaborar en la edición de los objetos del usuario.

2 Especificación Complementaria

2.1 Historia de revisiones

Versión del Documento	Fecha	Descripción de la Versión	Autores
1.0	24 /01/2012	Inicio proyecto	Jimena P. Alcántara Olivares José Daniel López Jaimes

Versión del Documento	Fecha	Descripción de la Versión	Autores
2.0	24 /01/2012	Corrección de reglas de dominio	Jimena P. Alcántara Olivares José Daniel López Jaimes

Versión del Documento	Fecha	Descripción de la Versión	Autores
3.0	10/07/2012	Corrección de reglas de dominio	Jimena P. Alcántara Olivares José Daniel López Jaimes

2.2 Introducción

En este documento se describen los requerimientos que imponen restricciones al diseño o al funcionamiento de la aplicación y que ayudarán a establecer las reglas del dominio. Se tomarán en cuenta los atributos que se deben de satisfacer para el desarrollo de un software (modelo FURPS), los cuales se explicarán a continuación.

2.3 Requerimientos no funcionales

FURPS acrónimo de (*Functionality, Usability, Reliability, Performance, Supportability*) o (Funcionalidad, Usabilidad, Fiabilidad, Rendimiento, Soporte) son un conjunto de factores de calidad de software. En la siguiente tabla se describe cada factor y como la aplicación ShareDraw respetará estos requerimientos.

[2] La tecnología multitouch consiste en una pantalla táctil que reconoce simultáneamente múltiples puntos de contacto, así como el software asociado a esta que permite interpretar dichas interacciones simultáneas.

REQUERIMIENTOS NO FUNCIONALES		
Requerimiento	**Descripción**	**Aplicación**
Funcionalidad	Se refiere a las características y capacidades del programa, la generalidad de las funciones entregadas y la seguridad del sistema global.	La aplicación brindará dos espacios de trabajo. Uno en el que el usuario será capaz de dibujar y editar figuras y texto sin la necesidad de conectarse a ningún otro dispositivo y el otro donde puede conectarse a un grupo de trabajo y compartir los objetos que cree y edite sabiendo en todo momento en que espacio se encuentra.
Usabilidad	Se valora considerando factores humanos, estética, consistencia y documentación general.	La aplicación será intuitiva con un diseño estético y fácil de usar, especialmente si se tienen antecedentes de aplicaciones de dibujo. Para la parte de conexión a un grupo de trabajo podría haber problema, aunque el usuario siempre tendrá una descripción de lo que está haciendo. De cualquier forma se contará con un pequeño tutorial de ayuda accesible desde la vista principal así como en cada espacio de trabajo.
Fiabilidad	Se evalúa midiendo la frecuencia y gravedad de los fallos, la exactitud de los resultados, el tiempo medio entre fallos, la capacidad de recuperación a fallos y de predicción del programa.	El usuario recibirá **retroalimentación** de la aplicación ya sea para saber que la aplicación está cargando, en que espacio de trabajo se encuentra o que herramienta u opción de edición está seleccionando. Los resultados que se esperan tienen que mostrarse en un tiempo de respuesta muy corto, incluso en el espacio colaborativo y se manejarán las excepciones y fallos con mensajes o acciones emergentes. Esto tomando en cuenta que aun cuando la aplicación tenga baja probabilidad de fallos debe estar preparada ante cualquier situación.
Rendimiento	Se mide por la velocidad de procesamiento, el tiempo de respuesta, consumo de recursos, rendimiento efectivo total y eficacia.	La aplicación estará programada pensando en el mejor aprovechamiento de los recursos del dispositivo.
Soporte	Combina la capacidad de	La aplicación hará uso de un servicio para

	ampliar el programa, adaptabilidad y servicios, así como capacidad de hacer pruebas, compatibilidad, capacidad de configuración, la facilidad de instalación de un sistema y la facilidad con que se pueden localizar los problemas.	la interconexión de los dispositivos, mismo que deberá estar previamente instalado. Será instalada como cualquier otra aplicación Android, ya sea con el único archivo de instalación ***apk*** o desde el centro de software de Android, estará pensada para tres posibles **resoluciones** y tamaños para ejecutarse en tabletas y celulares.

REQUISITOS ADICIONALES		
Requerimiento	**Descripción**	**Aplicación**
Implementación	Recursos de Hardware, lenguajes de programación y herramientas.	• PC con S.O. Windows (xp, vista, 7) o Linux(distribución utilizada Ubuntu 11.10 como máximo debido a restricciones para instalación de ciertos programas) • Samsung Galaxy S GT-I9000T (Android 2.3.3) • Samsung Galaxy Tab GT-P1000L (Android 2.3.6) • Xperia MiniPro SK17A (Android 2.3.4) • Java • XML 1.0 • Android SDK • Eclipse IDE • AVD Manager • Plugin ADT para Eclipse • Alljoyn SDK • Inkscape La aplicación será compatible y estable en dispositivos con Android 2.2 en adelante (al momento de la realización de este proyecto, la última versión es 4.0 *Ice Cream Sandwich*).
Interfaz	Restricciones a nivel de interacción con otros sistemas.	El sistema interacciona con el servicio Alljoyn en el módulo de interconexión del grupo de trabajo para la creación de

		la red.
Legales	Licencias, uso de hardware para la construcción de la aplicación.	No será necesaria ningún tipo de licencia ya que las herramientas en las que se programará a excepción del S.O. Windows son software libre.

2.4 Reglas del dominio

REGLAS DEL DOMINIO			
ID	**Regla**	**Grado de variación**	**Fuente**
REGLA1	El usuario siempre sabe en qué espacio de la aplicación se encuentra	El usuario puede cambiar entre espacios de trabajo constantemente, debe haber retroalimentación acerca de en qué espacio se encuentra	Estándares de diseño para aplicaciones Android
REGLA2	El usuario podrá entender todas las herramientas y a usar la aplicación fácilmente.	Se contará con distintas herramientas que podrían no ser, a primera instancia del todo intuitivas. Es necesario explicar al usuario qué está haciendo.	Definido por las funciones y herramientas que ofrece la aplicación
REGLA3	Se podrá cambiar entre espacios sin perder los datos de cada espacio.	Al cambiar entre los espacios debe mantenerse la información real de los espacios	Documento de propuesta de proyecto terminal
REGLA4	Los cambios deben darse en todos los dispositivos interconectados sin excepción y de manera eficiente.	Punto que debe respetarse estrictamente ya que se debe otorgar velocidad en los tiempos de respuesta.	Definido por las posibles fallas y cambios constantes que pudieran darse.

2.5 Información de dominios de interés

- **Aplicaciones Android**

  Dado que la aplicación será desarrollada para el S.O. Android, deben tomarse en cuenta los estándares y los consejos de diseño y programación de los desarrolladores profesionales. Android Developers[3] ofrece: reglas, ejemplos, explicaciones y guías prácticas para el desarrollo de aplicaciones.

- **Software colaborativo**

[3] http://developer.android.com/index.html

En la actualidad se ha dado un auge importante en las aplicaciones de tipo *groupware*[4] para trabajar de manera colaborativa en un proyecto. La aplicación SHAREDRAW tiende ser *groupware,* ya que se trabaja de manera cooperativa dentro de una red para realizar un trabajo (en este caso un dibujo). A futuro, este proyecto o la reutilización de alguno de sus módulos, puede servir para elaborar una aplicación más orientada a un modelo empresarial, que es la tendencia de aplicaciones groupware.

- **Aplicaciones de Dibujo**

  Un fuerte referente en cuanto a los antecedentes que podría haber de aplicación de dibujo es Paint[5] de Microsoft o MacPaint .

- **Conexión entre dispositivos**

  La conexión entre los dispositivos que compartirán objetos e información será a través de una red peer-to-peer. Para la comunicación p2p se usara Alljoyn[6] una framework que se encarga de reconocer las señales de cuando se conecta o desconecta un dispositivo y controlar los mensajes o datos que se comparten.

3 Glosario

Aplicación colaborativa	Referente a una aplicación que es ejecutada en varios dispositivos para trabajar en equipo
Groupware	Software que promete incrementar la productividad, ofreciendo a los usuarios una interfaz común con la que tienen acceso a una variedad de programas para trabajar de distintas maneras
Tareas Robustas	Tareas que gastan recursos considerables como memoria, procesamiento de operaciones, espacio de almacenamiento, interacción con otras aplicaciones como bases de datos y conexiones a redes inalámbricas.
P2P	Acrónimo de peer-to-peer un sistema distribuido de comunicación donde cada nodo puede compartir sus recursos solicitando y ofreciendo servicios dinámicamente
Portabilidad	Característica que posee un software para ejecutarse en distintas plataformas.
Vista gráfica	Referente a lo que se muestra al usuario en la pantalla del dispositivo.
Diseño Estético	Interfaz agradable basada en principios de diseño
APK	Un archivo con extensión .apk es un paquete para el sistema operativo Android. Este formato es una variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android para smartphones y

[4] http://es.wikipedia.org/wiki/Software_colaborativo
[5] http://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/mspaint_overview.mspx?mfr=true
[6] https://www.alljoyn.org/

	tablets.
Resolución	Formalmente resolución de pantalla, es el numero de pixeles que será mostrado en la pantalla
Retroalimentación	Se refiere a mensajes de vuelta cuando el usuario introduce una entrada para mantener al usuario al tanto de lo que sucede en la aplicación.
Gestos	Movimiento de la mano que expresa algo, en nuestro ámbito se refiere a los ademanes que alteran el comportamiento de algún objeto en Android. Ejemplo: el conocido "pellizco" para agrandar o hacer más pequeño un texto.
Espacio público/privado	Referente a las pantallas que se muestran al usuario y a los menús información y vistas que ofrecen dichas pantallas. El espacio público es aquel en el que el usuario comparte información a través de la red y el espacio privado es donde él puede realizar sus dibujos y únicamente su dispositivo puede ver lo que hace (hasta que lo comparta).
MultiTouch	La tecnología multitouch consiste en una pantalla táctil que reconoce simultáneamente múltiples puntos de contacto, así como el software asociado a esta que permite interpretar dichas interacciones simultáneas.

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Manual Técnico

Proyecto Terminal

Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android

Jimena Patricia Alcántara Olivares
Matricula: 207333058

José Daniel López Jaimes
Matricula: 207333480

Asesor: Dra. María Lizbeth Gallardo López
Profesor Investigador
Departamento de Sistemas

Trimestre 12O

Enero-2013

Índice

1 Hardware

Para la implementación de la aplicación *ShareDraw* se utilizó el siguiente *Hardware.*
Equipos

- Computadora de escritorio *Dell Studio XPS* 8100 Procesador *Intel core* i5 645, 4GB de memoria *RAM, S.O. Windows y Linux.*
- Computadora portátil *Dell Inspiron* 1545 Procesador *Intel Celeron*, 3GB de memoria *RAM, S.O. Windows y Linux*.

Dispositivos de Experimentación y Pruebas

- *Smartphone Samsung I9000 Galaxy*
  - Pantalla *WVGA* de 4 pulgadas
  - Procesador de 1 *GHz*
  - Sistema operativo *Android 2.1*
  - *Wi-Fi*

- *Tablet Samsung Galaxy GT-P1000L*
  - 10.1 pulgadas
  - Sistema operativo *Android 2.3.6.*
  - Procesador de doble núcleo a 1 *GHz.*

- *Smartphone Xperia MiniPro*
  - 3 pulgadas
  - Sistema Operativo *Android 2.3.4*
  - Procesador *Snapdragon* de 1 GHz

2 Software

Software utilizado para la implementación de la aplicación y para la realización de pruebas.

2.1 Sistema Operativo Android

Android es una solución completa de software de código libre para teléfonos y dispositivos móviles. Es un paquete que se compone de un conjunto de bibliotecas de bajo nivel, como son: *OpenSSL, SQLite y Webkit*; y de medio nivel, como son: *Content Providers, Activity Manager, Window y Manager*. Así mismo contiene un conjunto inicial de aplicaciones destinadas al usuario final.

Las aplicaciones *Android* están programadas en *Java,* siendo ejecutadas sobre *Dalvik,* una máquina virtual de Java desarrollada por *Google* y diseñada para optimizar la memoria y los recursos de hardware en los dispositivos portátiles.

Android se distribuye bajo una licencia libre permisiva que permite la integración con soluciones de código propietario.

2.2 AllJoyn Framework

Alljoyn es un marco tecnológico abierto de desarrollo para establecer una comunicación adecuada entre dispositivos próximos sin la necesidad de recurrir a un servidor intermediario, el cual se encuentra en una fase inicial de desarrollo. Actualmente, tiene en el mercado de *Android* solo unas 5 ó 6 aplicaciones funcionales.

Alljoyn ha sido diseñado para resolver las problemáticas que se presentan en la actualidad en el manejo de redes *P2P*, ofreciendo una solución a temas como: la detección transparente de dispositivos y servicios, funcionalidades de red, y el direccionamiento de mensajes; ofrece también un marco de seguridad con mecanismos de cifrado y autenticación de comunicaciones; además, busca facilitar la detección entre dispositivos, garantizando tasas reducidas de latencia.

2.3 Android SDK

Android SDK[1] es un kit de desarrollo que provee las bibliotecas y las herramientas de desarrollo necesarias para construir, probar y depurar aplicaciones para *Android*. Nosotros empleamos las versiones tanto para *Windows* como para *Linux*.

2.4 Entornos de Desarrollo

2.4.1 Eclipse Indigo (Java)

Eclipse Indigo es un entorno de desarrollo integrado (*IDE, Integrated Development Environment*) multiplataforma de código abierto que facilita las tareas de edición, compilación, ejecución y depuración de programas. Éste fue el principal entorno de desarrollo utilizado, debido a que aporta un plugin (*ADT plugin*) para *Eclipse* que extiende la funcionalidad de éste y facilita el desarrollo de aplicaciones para *Android*. Entre las funcionalidades de este *plugin* se encuentra:

- Emulador de *Android.* Permite elegir entre distintos terminales móviles y la versión del sistema operativo (ver figura 1).

[1] Del inglés *Standard Edition Kit*

- Editores de código para los distintos archivos de configuración (*XML*) que facilitan su comprensión y desarrollo (ver figura 2).
- Interfaces gráficas que permiten el desarrollo de componentes visualmente (ver figura 3).
- El acceso a herramientas de desarrollo de *Android* como tomar capturas de pantalla, la posibilidad de depurar con puntos de parada o ver el estado de los hilos y los procesos corriendo en el sistema).

Figura 1. Vista del emulador.

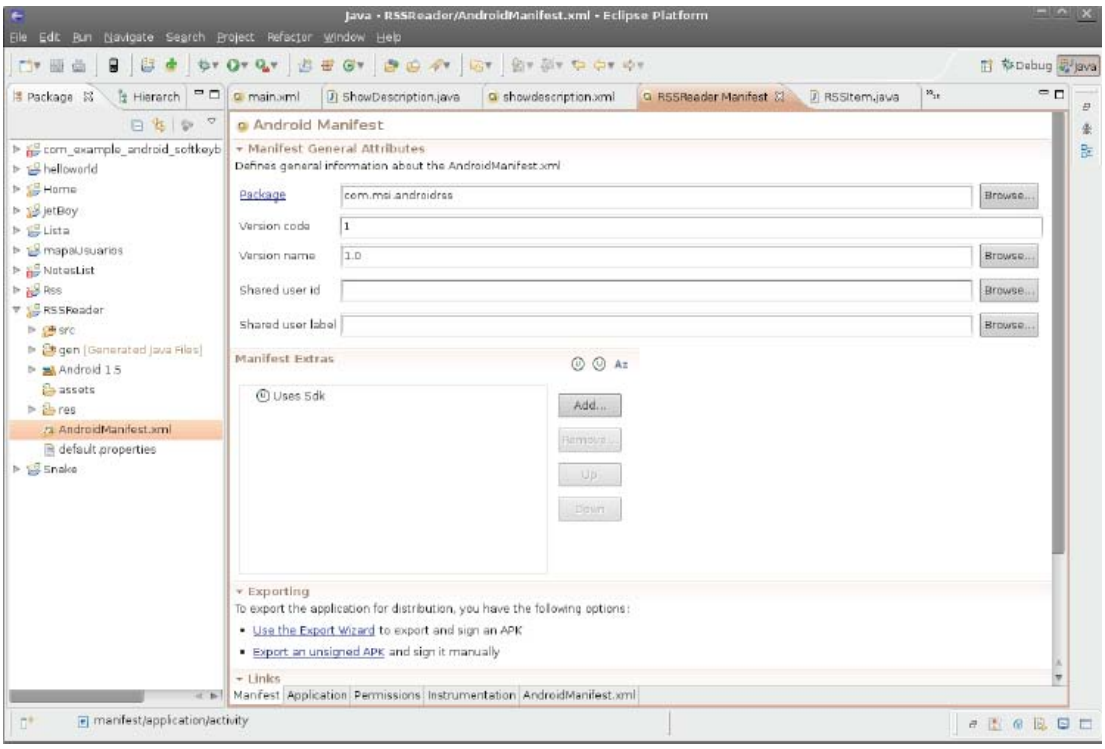


Figura 2. Editor de Código XML

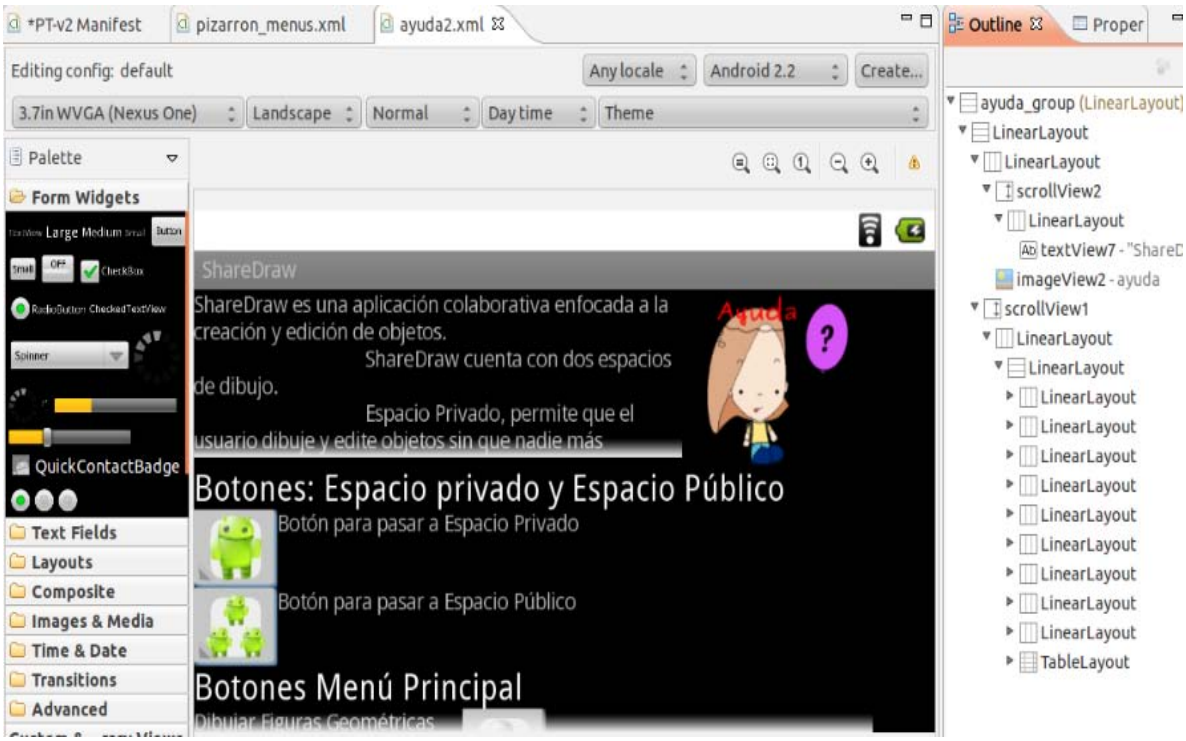


Figura 3. Interfaz Gráfica de desarrollo de componentes.

2.4.2 Netbeans 7.0.1

Netbeans IDE es un entorno de desarrollo libre escrito en *Java* que permite escribir, compilar, depurar y ejecutar programas. La plataforma *NetBeans* permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de *java* escritas para interactuar con las *API* de *NetBeans*, lo cual permite extender las propiedades del mismo *NetBeans*. Por tanto este entorno también permite hacer una integración de desarrollo para crear aplicaciones de Android.

3 Aplicaciones finales del Proyecto Terminal

Es importante aclarar que el módulo colaborativo no está integrado a la aplicación *ShareDraw*, ya que el *framework Alljoyn* es de reciente creación, la documentación no es precisa en cuanto a cómo utilizarla y no cuenta con ejemplos suficientes. Por lo tanto, se experimentó por separado y se creó un módulo independiente llamado *ModuloColaborativoShareDraw*. Este módulo se implementó para dos arquitecturas: Cliente-Servidor y P2P, ofreciendo como funcionalidades: i) que los dispositivos móviles se unan a una sesión que uno de ellos inicie previamente; y ii) la edición: mover, cambiar el tamaño y cambiar el color de una sola figura, la cual es creada por el mismo módulo de colaboración.

4 Estructura del Directorio de las aplicaciones ShareDraw y ModuloColaborativoShareDraw

Una vez instalados todos los paquetes de desarrollo es importante conocer la jerarquía del directorio de una aplicación de *Android,* la cual es la misma para *ShareDraw y ModuloColaborativoShareDraw* (ver Figura 4), las carpetas que contienen son las siguientes:

src/: La carpeta de contenido (*source folder*) incluye todas aquellas clases de *java* generadas para el funcionamiento de la aplicación *ShareDraw y ModuloColaborativoShareDraw.*

gen/: contiene archivos de Java generados por ADT *plugin*. El *ADT* crea un archivo `R.java`, el cual contiene las referencias a cada uno de los recursos de la carpeta *res*, es mediante este archivo que se pueden invocar los recursos en la clases requeridas.

Android version/ incluye el archivo `Android.jar` que se construyó a partir de la versión 2.2 de *Android*, la cual fue seleccionada cuando se creó el proyecto para desarrollar *ShareDraw y ModuloColaborativoShareDraw.*

res/: La carpeta recursos (*resource*) contiene los diversos recursos que *ShareDraw y ModuloColaborativoShareDraw* consumen, subdividiéndolos en carpetas dependiendo del tipo de archivo. Los distintos tipos de archivos que consume nuestra aplicación son *anim* dirigido a animaciones para las vistas, *drawable* contiene aquellas imágenes utilizadas en la aplicación, *drawable-hdpi* contiene imágenes para los dispositivos con pantalla de alta definición, *drawable-ldpi* contiene imágenes para los dispositivos con pantalla de baja definición, *drawable-mdpi* contiene imágenes para los dispositivos con pantalla de definición media, *layout* contiene archivos *XML* que definen las interfaz gráfica, *values* contiene archivos *XML* que contienen *Strings*.

Android.Manifest.xml: éste es un archivo *XML* requerido para cualquier aplicación de Android, está localizado en la raíz del directorio del proyecto, y su función es describir los valores globales del proyecto, incluyendo los componentes de la aplicación (actividades, servicios, etc.). Es importante mencionar que al declarar un componente, se declaran al mismo tiempo algunos de sus atributos como son: el momento de iniciar el componente, la posición o el tema que adoptará una vez iniciado el componente.

Figura 4. a) Jerarquía de las carpetas del Proyecto, b) carpeta libs.

Finalmente *ModuloColaborativoShareDraw* contiene una carpeta más, necesaria para implementar la colaboración en conjunto con el *framework Alljoyn.*

libs/: esta carpeta se compone de dos paquetes de librerías de *Alljoyn: Alljoyn.jar y liballjoyn_java.so (ver figura 4 inciso b).*

Clases que conforman la Aplicación ShareDraw.

ManejadorEspacios.

Esta clase hereda los atributos y métodos de la clase *Activity*. Por lo tanto la clase *ManejadorEspacios* permite construir la interfaz tanto del espacio público como del espacio privado de la aplicación ShareDraw.

EspacioPrivado

Esta clase está compuesta por una clase *View*, la cual permite mostrar la vista del *MenuPrincipal*, la del *EspacioPrivado* y la del *MenuObjeto*.

EspacioPublico

Esta clase está compuesta por una clase *View*, la cual permite mostrar la vista del *MenuPrincipal*, la del *EspacioPublico* y la del *MenuObjeto*.

ShareDrawListener

La clase *ShareDrawListener* tiene como función principal seleccionar las acciones a realizar en la aplicación. Es importante señalar que estas acciones dependen del espacio en el que se realice la acción (ver Tabla 1). Las acciones son:

Espacio Privado	Acciones	Espacio Público
✓	Crear figura	✓
✓	Crear Texto	✓
✓	Conectarse a una red para colaborar con otros usuarios.	X
✓	Manual de ayuda breve.	✓
✓	Asignar Color a figura o texto.	✓
✓	Compartir figura o texto del Espacio Privado al Público.	X
X	Importar figura o texto del Espacio Público al Privado.	✓
✓	Duplicar figura o texto.	✓
✓	Borrar figura o texto.	✓

✓	Editar texto.	✓

Tabla 1. Acciones posibles a realizar en los espacios respectivos.

La acción conectarse a una red para colaborar con otros usuarios solo puede ser solicitada por un usuario que está trabajando inicialmente en su espacio privado; la acción compartir figura o texto del Espacio Privado al Público solo puede ser solicitada por el usuario que desea compartir uno o varios objetos desde su espacio privado; mientras que la acción Importar figura o texto del Espacio Público al Privado solo puede ser solicitada por un usuario que desee modificar una copia de los objetos que se encuentran en el espacio público.

DrawFrame

DrawFrame tiene como función principal el crear los objetos gráficos o eliminarlos, así como la edición de éstos en su tamaño, color o contenido.

ObjetoGrafico

ObjetoGrafico es una clase genérica que contiene los métodos y los atributos que comparten las clases: *Poligono, PoligonoSimple* y *Texto*.

Poligono

Poligono es una clase que hereda de *ObjetoGrafico* y su función es crear polígonos uniformes; es importante destacar que no existen métodos de creación directa de un polígono en los paquetes de gráficos de *Android;* por lo tanto se realizó un algoritmo para dibujar polígonos.

PoligonoSimple

PoligonoSimple es una clase que hereda de *ObjetoGrafico* y su función es dibujar figuras simples: línea, círculo, cuadrado y rectángulo.

Texto

Esta clase hereda de *ObjetoGrafico*. Texto es un objeto que contiene una cadena de texto o mensaje, el cual es el que se desea dibujar.

Clases que conforman la Aplicación *ModuloColaborativoShareDraw*

`PTInterface`

`PTInterface` es una interface necesaria para definir el modelo lógico dentro del sistema distribuido de ShareDraw; describe el comportamiento y los métodos (señales) que serán invocados de manera remota.

`AlljoynHandler`

`AlljoynHandler` es una clase que hereda de `Handler` (Clase de *Alljoyn*). Es mediante esta clase que se realizan todas las llamadas al bus de comunicación, como son: registrar objetos *(BusObject*), *listener's* [2] (*SessionListener y PortListener*) y *handlers* [3] (*SignalHandler*). Así mismo permite conectarse, desconectarse del bus, crear sesiones y desconectarse de ellas.

`Colaboration`

`Colaboration` es una clase que hereda de Activity y su función es detectar los cambios que debe comunicar al bus de comunicación, lo cual es realizado mediante el objeto `AlljoynHandler` creado en esta clase.

`MCPT`

`MCPT` es una clase que hereda de *Activity,* su función es presentar un menú de tres opciones al usuario para crear o unirse a una sesión colaborativa o cancelar la opción de colaborar con más usuarios.

`PTService`

`PTService` es una clase que implementa a la interfaz `PTInterface`, así como a la interfaz `BusObject` (perteneciente a *Alljoyn*). Esta clase es necesaria para la clase `AlljoynHandler.`

`PTBusListener`

`PTBusListener` es una clase que hereda de `BusListener` (perteneciente a *Alljoyn*). Esta clase es responsable de manejar aquellos eventos del Bus de comunicación.

[2] *Listener*: la teoría establece que existe un objeto al que llamaremos disparador que realiza alguna acción sobre otro objeto reactivo. El objeto entonces informa a varios *Listener* de que algo ha ocurrido en el disparador. Entonces cada uno de los *listeners* puede realizar alguna acción en base al evento que ocurrió.

[3] *Handler*: manejador de señales.

5 Proceso de instalación del software necesario para implementar

5.1 Instalación Android SDK

Descargar del siguiente enlace http://developer.android.com/sdk/index.html la última versión del *SDK* para la plataforma en la que se desarrollará (*Windows, Linux, MacOSx*). Para este proyecto las versiones elegidas fueron:

- *installer_r20.0.3-windows.exe*
- *android-sdk_r20.0.3-linux.tgz*

En este caso se explicarán los pasos para instalarlo en un ambiente Linux, ya que el framework Alljoyn solo tiene compatibilidad con la versión 2.3 de *Android* o anteriores.

1) Descomprimir *Android SDK* en alguna carpeta del directorio del sistema operativo.
2) Declarar el *path* para que el sistema encuentre los ejecutables. Para ello hay que editar el fichero *".bashrc"*. En una consola se pone la siguiente instrucción.

   $ gedit ~/.bashrc

3) Añadir al final del fichero la siguiente línea

   export PATH=${PATH}:/home/{usuario}/{carpeta}/android-sdk-linux_86/tools

4) Guardar los cambios.
5) Introducir en consola
   $ Android

Esto hace que se ejecute el manager de *Android* donde podremos instalar y actualizar el *SDK* así como crear dispositivos virtuales de pruebas, como se muestre en la figura 5.

Figura 5. Manager de Android.

6) Se selecciona *Available Packages* y después pulsar el botón *refresh.*

7) A continuación aparecerán todos los paquetes que se deseen instalar conteniendo diversas versiones del *SDK* de *Android*, ver figura 6.

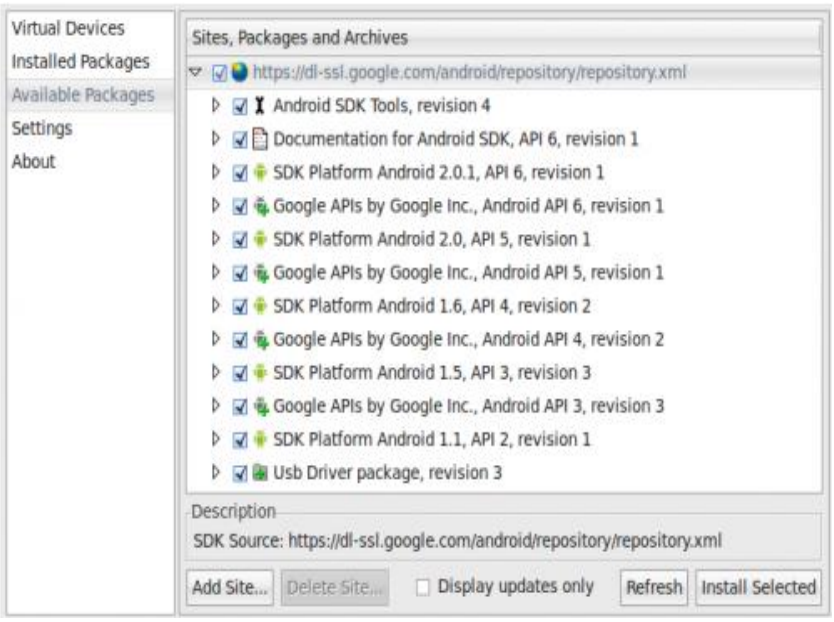


Figura 6. Manager Android Paquetes de Distintas versiones del SDK

Para esta aplicación se utilizó el *SDK Platform Android 2.0, API 5*.

8) Una vez seleccionada la versión y pulsado el botón *Install Accepted* aparecerá una ventana para aceptar las licencias de dichos paquetes, se da aceptar y como siguiente paso se instalará el *SDK* de *Android*.

5.2 Eclipse

En este caso se utilizó la versión *Eclipse Indigo*, para descargarlo ir al siguiente link http://www.eclipse.org/downloads/packages/release/indigo/sr2

1) Descargar *Eclipse IDE* para *Java EE Developers*.
2) Eclipse no necesita instalación, simplemente se descomprime el fichero descargado.
3) Ejecutar *Eclipse* para configurarlo.

5.3 Instalar ADT plugin de Android.

1) Seleccionar del menú de herramientas de Eclipse, la opción *Help>Install New Software.*
2) A continuación pulsar el botón *Add*. Aparecerá una ventana.
3) Escribir los siguientes datos donde corresponda, como se observa en la figura 7.

Name: Android Plugin
Location: https://dl-ssl.google.com/android/eclipse/

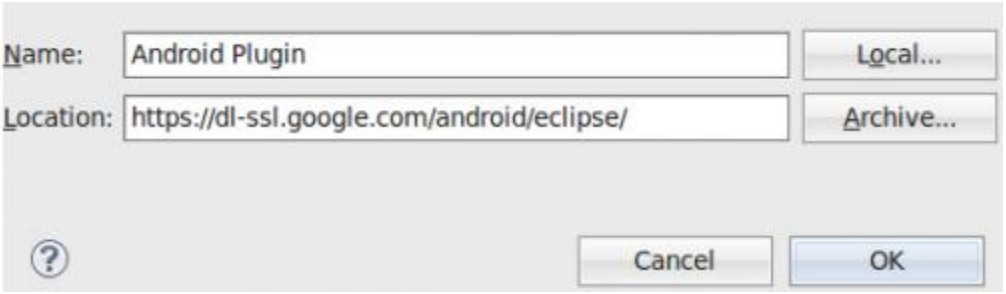


Figura 7.Dar la url del Android Plugin

4) Pulsar OK, se mostrará la siguiente ventana, ver figura 8.

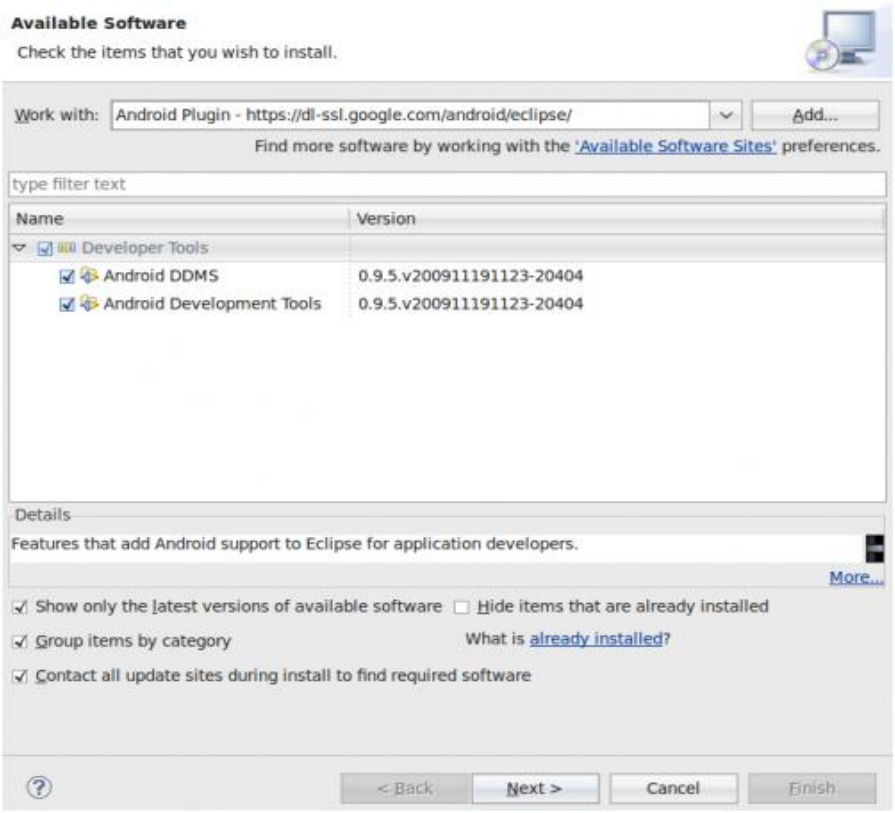


Figura 8. Herramientas a instalar del plugin

5) Se seleccionan las dos opciones y se da siguiente.
6) Se aceptan los términos de las licencias y se pulsa *Finish.* Entonces empezará a descargar todo lo necesario para que funcione el plugin, ver figura 9.

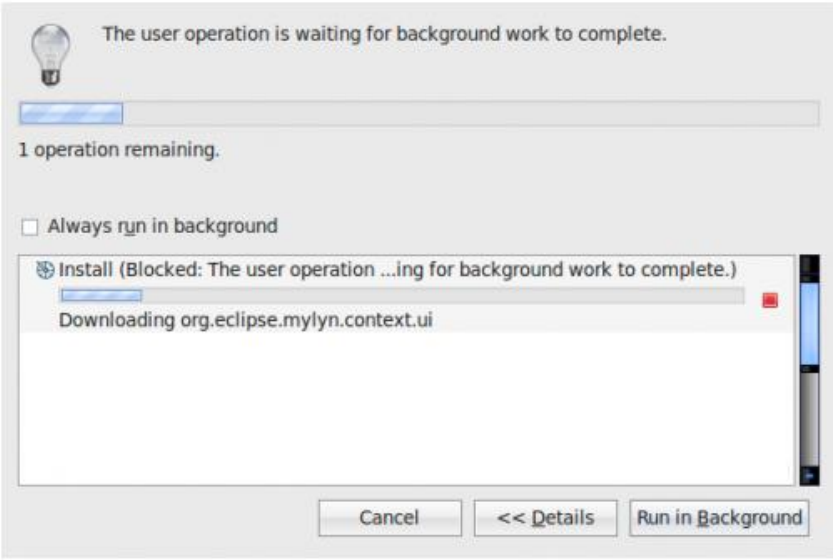


Figura 9.Instalación del ADT Plugin

7) Por último se debe indicar dónde está el *Android SDK,* para esto se selecciona la opción *window* en el menú de herramientas, *window>preferences>Android.*

8) Se pulsa el botón *Browse* y colocamos la dirección donde se descomprimió el paquete del *SDK.*
9) Finalmente aparecerá la lista de todo lo que hemos instalado como se muestra en la figura 10.

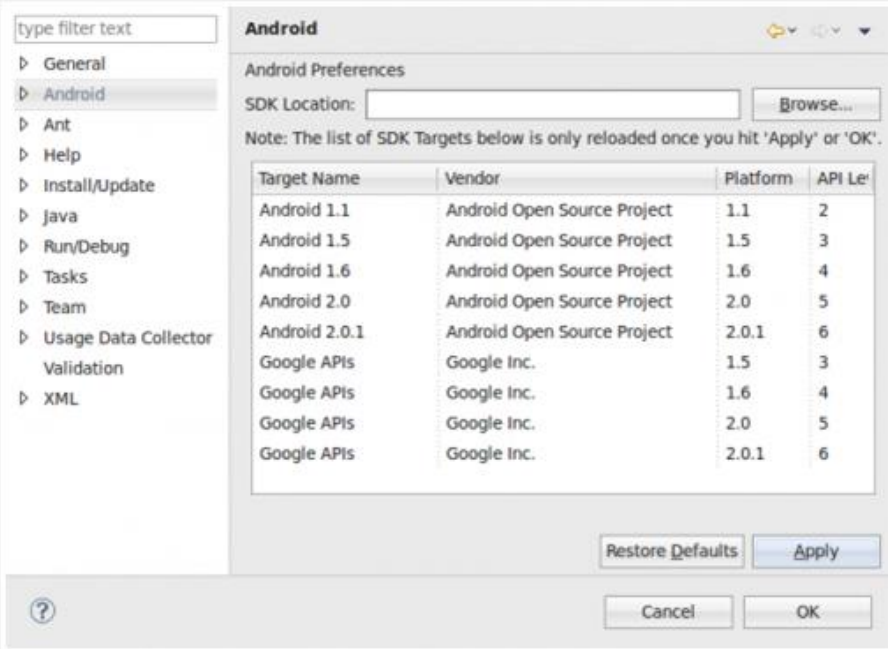


Figura 10.Lista de paquetes instalados.

5.4 Alljoyn

Configuración del entorno de programación para el desarrollo de aplicaciones de Alljoyn con Android.

Seguir las instrucciones de configuración del siguiente enlace:
https://www.alljoyn.org/docs-and-downloads/documentation/alljoyn-android-environment-setup-guide#unique_10

Configuración del entorno de desarrollo para la plataforma de Linux.

Seguir las instrucciones de configuración del siguiente enlace:
https://www.alljoyn.org/docs-and-downloads/build-environment/configuring-build-environment-linux-platform

Una vez instaladas las tecnologías descritas se podrán cargar los dos proyectos: *ShareDraw y ModuloColaborativoShareDraw* en *Eclipse* para integrarlas y continuar su implementación.

Un aspecto importante a destacar es la integración del módulo colaborativo con la aplicación ShareDraw. Para lograrlo, es necesario crear un método en la clase `AlljoynHandler` para cargar todos los objetos gráficos de la sesión en la lista de objetos cada vez que un usuario o se una a la sesión. También es necesario crear una clase que se encargue de dividir el mensaje de la señal enviada entre los dispositivos y que decida qué objeto está siendo editado.

6 Proceso de Instalación de la Aplicación en dispositivo móvil

1) Ubicar archivos necesarios a instalar, existen tres *apk's*[4] a instalar en el dispositivo móvil:
   - *PT-v2.apk*: contiene la aplicación ShareDraw sin integrar el módulo colaborativo de la aplicación.
   - *Alljoyn.apk*: contiene el servicio Alljoyn necesario para el módulo colaborativo de ShareDraw.
   - *ModuloColaborativoShareDraw.apk*: contiene el módulo colaborativo de ShareDraw; trabaja en conjunto con Alljoyn.apk.

   Los dos primeros se encuentran en la carpeta *bin* del proyecto llamado PT-v2; el tercero se encuentra en la carpeta *bin* del proyecto llamado ModuloColaborativoShareDraw.
2) Copiar y guardar en la tarjeta de almacenamiento interna o externa del dispositivo móvil los tres *apk's* (para este paso es necesario conectar el dispositivo al equipo donde se encuentre los *apk's*).
3) Para instalar los apk's contenidos ahora en el dispositivo será necesario utilizar un explorador de archivos; en este caso se recomienda *Astro* (descargable desde Android Market).
4) Abrir *Astro,* ubicar los tres archivos apk en la memoria, seleccionar el apk se mostrarán tres opciones seleccionar instalar. Finalmente se deberá repetir este paso para cada apk. Una vez instalados los tres archivos, se podrán utilizar. En caso que se desee usar el módulo colaborativo de ShareDraw se deberá iniciar primero la aplicación llamada *Alljoyn*, después se podrá iniciar la aplicación *ModuloColaborativoShareDraw.*

[4] *Apk (Application Package File)* Las aplicaciones móviles para Android tienen la extensión .apk (variante del formato JAR de Java), la cual se usa para distribuir e instalar aplicaciones para la plataforma Android.

7 Pruebas

Las pruebas realizadas para implementar *ShareDraw* son:
Implementación de vistas (*View*) y diálogos con distintos tipos de layouts para definir así la interfaz gráfica deseada.
Implementación de distintas algoritmos para dibujar figuras.
Pruebas de implementación para comprender la tecnología Multitáctil.
Experimentación en la edición de los objetos gráficos para hallar la mejor forma en que podrían ser editados.

Las pruebas realizadas para implementar *ModuloColaborativoShareDraw* son:
Pruebas de conexión al servicio de *Alljoyn.*
Pruebas en base a dos ejemplos proporcionados por *Alljoyn.*
Implementación de un modelo de comunicación Cliente-Servidor mediante *RMI.*
Experimentación con señales para lograr un modelo de comunicación *P2P.*
Prueba del módulo colaborativo conectando tres dispositivos.

8 Problemas presentados en el desarrollo de la aplicación

En general presentó un reto importante el desarrollo de esta aplicación, ya que el *software* móvil tiene que satisfacer varias condicionantes como: adaptabilidad de la interfaz gráfica a diferentes tamaños de pantallas, riesgos de integridad de datos, soportar interrupciones de tareas, existencia de una considerable variedad de estándares como protocolos de comunicación, tecnologías de red diferentes.

Módulos	Solución
Módulo de Presentación ➢ Manejo de Actividades. ➢ Adaptabilidad de la Interfaz Gráfica	 ➢ Investigación del uso de las actividades. ➢ Diseño de la interfaz mediante principios de Android.
Módulo de Gestión de Objetos ➢ Manipulación de vistas ➢ Manejo de eventos multitáctiles sobre las vistas (objetos gráficos).	 ➢ Investigación en la documentación de Android ➢ Investigación de los eventos y de cómo reconocer los eventos.
Módulo de Transición entre espacios ➢ Manejo de integridad de los datos en el cambio de vistas	 Idear una navegabilidad entre las clases las cuales permitan instanciar los datos de una forma

	segura.
Módulo de Colaboración ➢ Utilizar un *framework* ajeno a los paquetes que distribuye *Android*. ➢ Falta de documentación para implementar código adaptable a la aplicación. ➢ Manejo de señales de comunicación con el bus de comunicación	Debido a la falta de tiempo y a la complejidad de la implementación de este módulo no se integró a la aplicación *ShareDraw; pero se* creó un módulo denominado *ModuloColaborativoShareDraw*, el cual permite colaborar en la edición de una figura geométrica. La figura geométrica podrá ser editada en su posición, tamaño y color por los usuarios que se unan a la sesión del módulo.

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Manual del Usuario

Proyecto Terminal

Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android

Jimena Patricia Alcántara Olivares
Matricula: 207333058

José Daniel López Jaimes
Matricula: 207333480

Asesor: Dra. María Lizbeth Gallardo López
Profesor Investigador
Departamento de Sistemas

Trimestre 12O

Enero-2013

Índice

1 Introducción

Existen dos aplicaciones que el usuario podrá utilizar *ShareDraw y ModuloColaborativoShareDraw*.

ShareDraw es una aplicación colaborativa enfocada a la creación y edición de objetos. Cuenta con dos espacios de dibujo:
Espacio Privado, permite que el usuario dibuje y edite objetos sin que nadie más colabore.
Espacio Público, permite que el usuario y otros usuarios editen de manera colaborativa los objetos del espacio. *ShareDraw* permite crear figuras geométricas, así como mensajes de texto breves. El usuario podrá compartirlos con otras personas estando conectados a una misma red y colaborar para su edición.

Las figuras y mensajes de texto podrán ser duplicados, eliminados o editados en su tamaño y color, además los mensajes podrán ser editados en su contenido.

La aplicación cuenta con dos menús, el menú principal contiene las siguientes opciones: creación de figuras, creación de mensajes de texto, manual de ayuda, en el espacio privado existe la opción de unirse a un grupo de colaboración; el cual activará el espacio público donde podrá trabajar con las otras personas.

Es importante aclarar que el módulo colaborativo no está integrado a la aplicación *ShareDraw* porque el *framework Alljoyn* es de reciente creación, la documentación no es precisa en cuanto a cómo utilizarla y no cuenta con ejemplos suficientes. Por lo tanto, se experimentó por separado y se creó un módulo independiente llamado *ModuloColaborativoShareDraw*. Este módulo se implementó para dos arquitecturas: Cliente-Servidor y P2P, ofreciendo como funcionalidades: i) que los dispositivos móviles se unan a una sesión que uno de ellos inicie previamente; y ii) la edición: mover, cambiar el tamaño y cambiar el color de una sola figura, la cual es creada por el mismo módulo de colaboración.

También cabe mencionar que los botones de *ShareDraw* relacionados con la funcionalidad de colaboración deberán adaptarse para integrar el módulo de colaboración. Asimismo, el Espacio Público se mantendrá activo permitiendo así el paso de objetos gráficos entre espacios, más adelante se comentarán estas funcionalidades (importar y compartir).

2 Utilizar la aplicación ShareDraw

Iniciar Aplicación, seleccionar el ícono de S*hareDraw*.
Aparecerán tres opciones a elegir en la pantalla inicial, como se observa en la Figura 1.

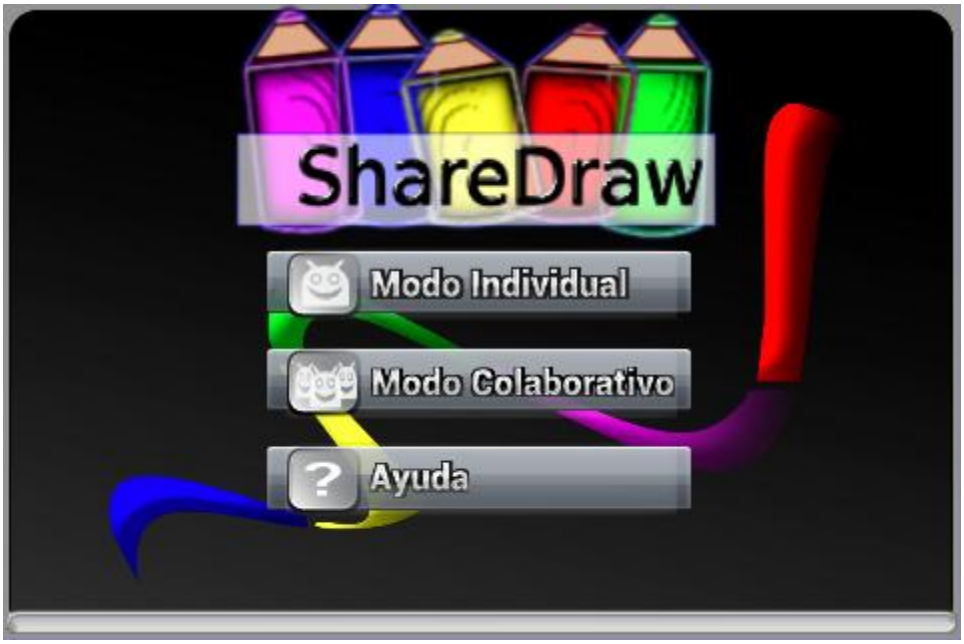


Figura 1. Pantalla Principal.

2.1 Menú principal.

En la figura 2 se muestra la vista del espacio Privado con su menú principal.

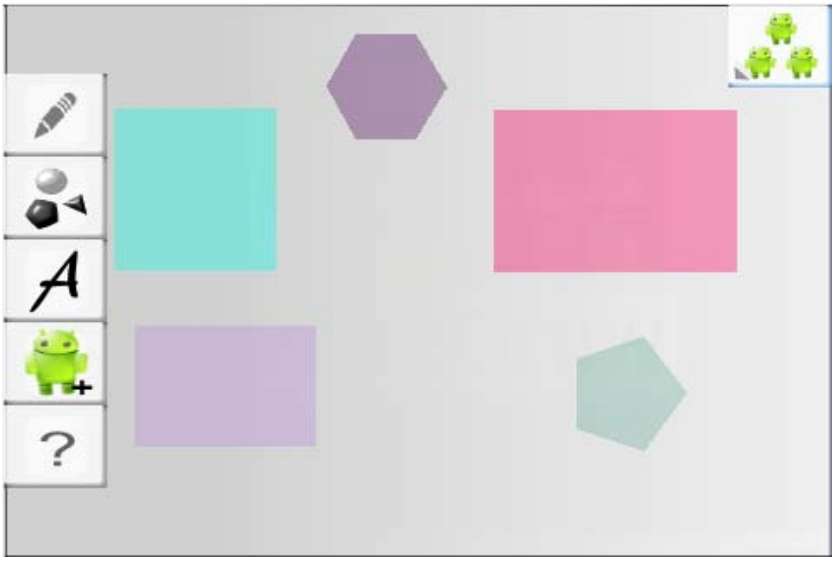


Figura 2. Espacio Privado y Menú Principal.

Opciones a realizar con menú principal

Crear figuras

Si se elige esta opción aparecerá un diálogo con las figuras que se podrán seleccionar (ver figura 3). Una vez pulsado la figura deseada es necesario que el usuario indique en qué posición del espacio de dibujo desea que aparezca.

Figura 3. Diálogo para crear figuras.

Crear Texto

Si se elige esta opción aparecerá un diálogo con una caja de texto donde se podrá introducir el mensaje que desee escribir el usuario, ver figura 4. El mensaje tendrá una limitación del tamaño de la cadena.

Figura 4. Diálogo de Crear Texto

Unirse a Dibujar en modo Colaborativo

Esta opción está pensada para compartir algún mensaje o figura con otro usuario. Esta opción está relacionada con el módulo colaborativo, pero por el momento está inhabilitada en *ShareDraw*. Para esta funcionalidad se tendrá que estar conectado a la misma red del otro usuario e inicializar el servicio de Alljoyn, lo cual se hace seleccionando el ícono de Alljoyn que se encuentra en el menú de aplicaciones del dispositivo. Una vez cumplidas las restricciones se activará el Espacio Público.

Ayuda

Si se elige esta opción aparecerá un diálogo que contiene un pequeño manual de las funciones de los botones y de las funcionalidades que ofrece la aplicación, en caso de que el usuario no recuerde alguna función podrá recurrir a éste, en la figura 5 se muestra el diálogo del manual.

Figura 5. Diálogo de Ayuda.

2.2 Botón de Cambio entre Espacios

Este botón permitirá cambiar entre espacios, en caso de que se encuentre en un modo colaborativo. El ícono del botón indicará a que espacio puede cambiar y en el que se encuentra.

Ir a Espacio Público 

Ir a Espacio Privado 

2.3 Menú Objeto

Para que el usuario pueda visualizar el menú objeto será necesario que toque a la figura a editar por más de un segundo y procurar que no exista movimiento de ésta. En caso de que cueste trabajo, debido a que se mueve demasiado la figura, se podrá utilizar más de un dedo y volver a tocar a la figura por más de un segundo, procurando poner los dos dedos al mismo tiempo y sin mover la figura.

Opciones a realizar con menú objeto.

Las opciones que ofrece el menú objeto cambian dependiendo del objeto seleccionado. Si el objeto seleccionado es una figura geométrica las funciones que presenta este menú son: editar color, importar, duplicar y eliminar (ver figura 6). En cambio, si el objeto es Texto las funciones que presenta este menú son: editar color, editar texto, importar, duplicar y eliminar (ver figura 7).

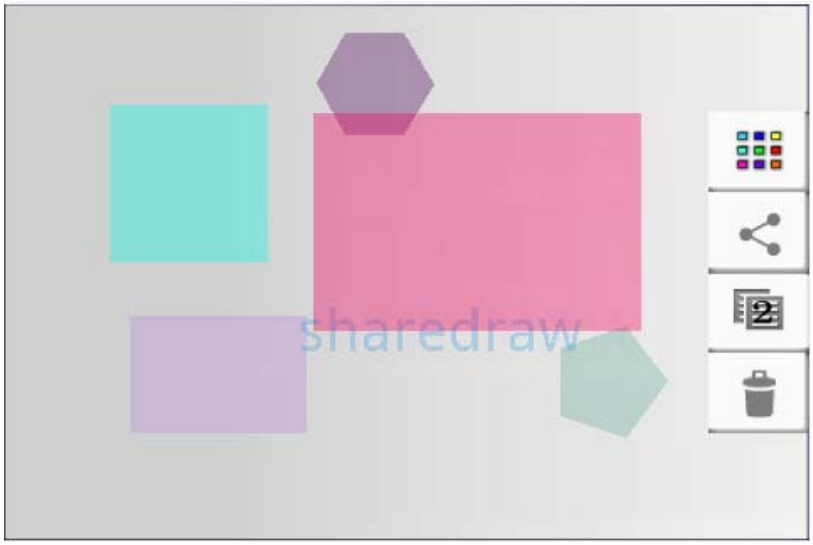


Figura 6. Menú objeto para una figura

Figura 7. Menú objeto para un objeto tipo Texto.

Cambiar Color.

Si se elige esta opción aparecerá un seleccionador de color. Lo único que se tiene que hacer es tocar el color deseado y confirmar el color dando un toque sobre el círculo central del seleccionador. En la figura 8 se muestra el dialogo seleccionador de color.

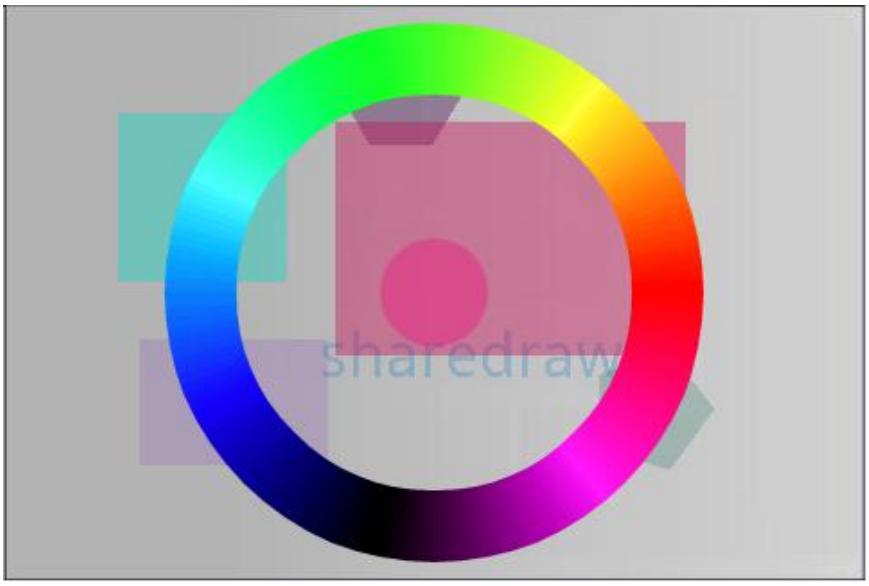


Figura 8. Seleccionador de color.

Duplicar

Esta opción si se pulsa duplicará el objeto gráfico seleccionado.

Eliminar

Esta opción eliminará el objeto gráfico seleccionado, si el usuario confirma la acción.

Importar

Para importar un objeto será necesario que el usuario se encuentre en el Espacio Público. Idealmente, la acción de importar consiste en copiar el objeto gráfico seleccionado al Espacio Privado, así el usuario podrá modificar individualmente el objeto, sin la intervención de los otros colaboradores. Actualmente este botón actúa únicamente sobre el Espacio Público instalado en el dispositivo del usuario; es decir, permite crear un objeto gráfico en el Espacio Público e importarlo al Espacio Privado local.

Compartir

Para compartir un objeto será necesario que el usuario se encuentre en el Espacio Privado. Idealmente, la función compartir consiste en copiar el objeto gráfico seleccionado al Espacio Público, así otros usuarios podrán colaborar en la edición del objeto. Actualmente este botón actúa únicamente sobre el Espacio Público instalado en el dispositivo del usuario; es decir, permite crear un objeto gráfico en el Espacio Privado y compartirla al Espacio Público.

Cambiar Tamaño

Para cambiar de tamaño una figura se hará un gesto multitáctil, el cual es utilizado en aplicaciones de dibujo o en la galería de imágenes de Android. Este gesto comúnmente se le conoce como pellizco y consiste en colocar dos dedos sobre el objeto gráfico a agrandar o reducir. Si se desea que el objeto crezca, los dedos se separan uno de otro, sin separarlos de la pantalla; si se desea reducir el objeto, los dedos deben acercarse tal y como se muestra en la figura 9.

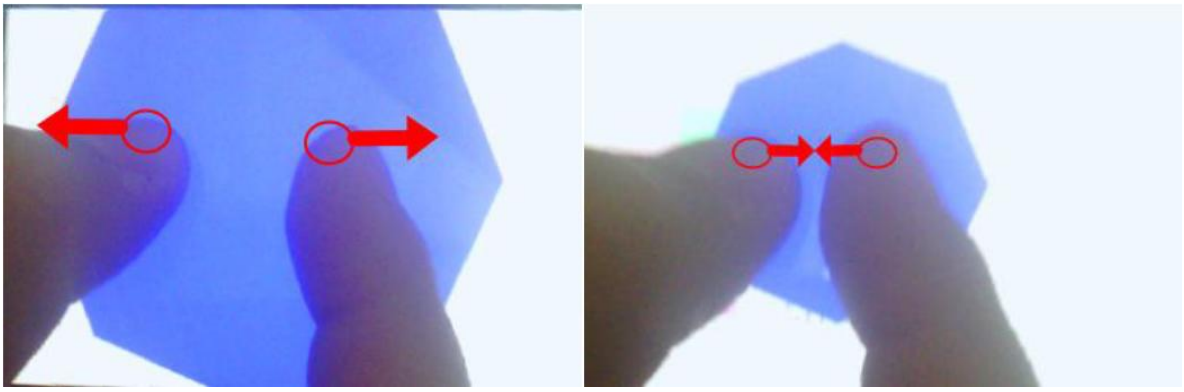


Figura 9. Cambiar tamaño por Gesto Multitáctil

3 Utilizar la aplicación *ModuloColaborativoShareDraw*

Iniciar Aplicación, seleccionar el ícono de *ModuloColaborativoShareDraw.* Aparecerán tres opciones, ver figura 10.

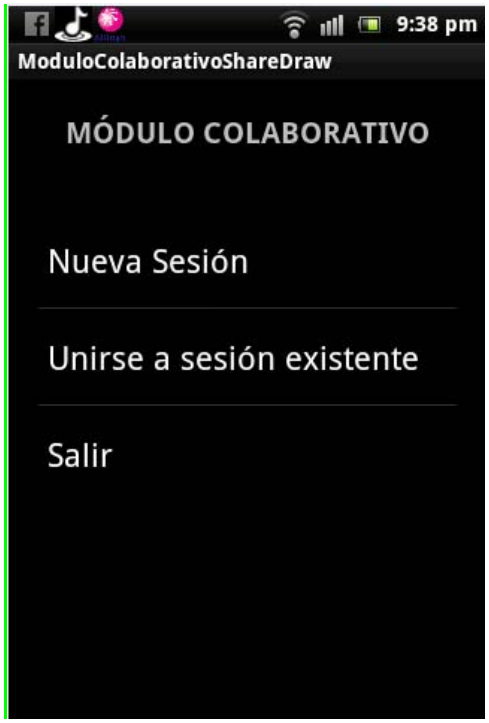


Figura 10. Pantalla Principal con tres opciones.

Si no existe una sesión de grupo de colaboración, el usuario deberá seleccionar la opción "*Nueva Sesión*", en la figura 11 se muestra la ventana siguiente, en donde el usuario definirá el nombre de la nueva sesión.

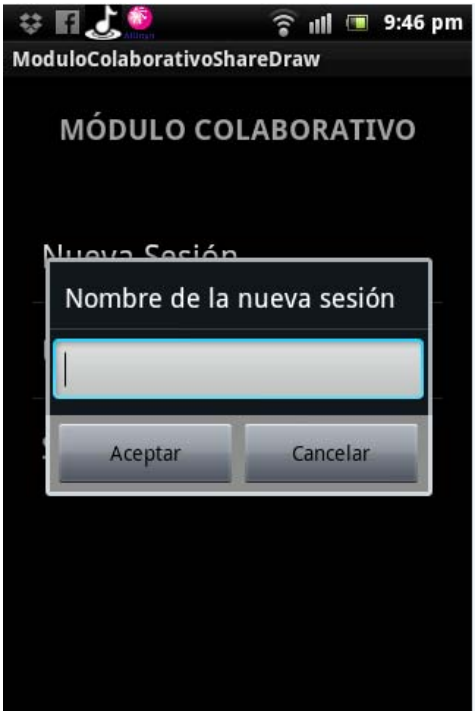


Figura 11. Diálogo Nueva Sesión.

En caso que exista una sesión de colaboración, el usuario deberá seleccionar la opción *"Unirse a sesión Existente*", en la figura 12 se muestra el diálogo donde el usuario tendrá que escribir el nombre de la sesión a la que desea unirse.

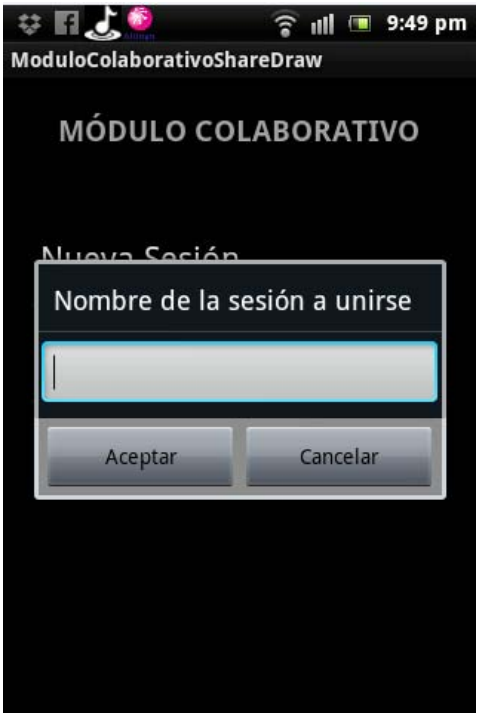


Figura 12. Diálogo Unirse a sesión existente.

A continuación se mostrará el espacio de dibujo en el que se encontrará una figura geométrica, la cual podrán editar en modo codo colaborativo los usuarios conectados a la sesión, ver figura 13.

Figura 13. Espacio de dibujo ModuloColaborativoShareDraw.

Las opciones a editar en la figura geométrica son: cambiar de posición (mover la figura), cambiar tamaño y cambiar color. Las dos primeras funciones se realizan de la misma forma que en *ShareDraw,* para el cambio de color será necesario que el usuario toque la figura por más de un segundo con tres dedos, procurando que no exista movimiento de ésta. En caso que se cumplan estos requisitos se visualizará el diálogo con el seleccionador de color antes mostrado, ver figura 8.

Alljoyn se encarga de manejar los errores principales de comunicación dentro de un sistema distribuido, por ejemplo: concurrencia, bloqueos, fallos en red, perdida de mensajes, saturación en el tráfico, entre otros. Así *ModuloColaborativoShareDraw* permite que los usuarios conectados puedan visualizar los cambios de la figura en tiempo real, como ejemplo: si un usuario se encuentra moviendo la figura los demás usuarios podrán observar el movimiento al mismo tiempo. También la aplicación permite que un usuario mueva una figura al mismo tiempo en que otro usuario se encuentre cambiando el color o el tamaño de la figura. Asimismo, si los usuarios se encuentran editando la misma opción, por ejemplo que dos usuarios editen el tamaño de una figura, la aplicación se encargará de manejar los cambios sin bloquear la edición de ésta, modificando la figura en base al orden en que se dieron los cambios.

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Memoria de Diseño

Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android

Jimena Patricia Alcántara Olivares
Matricula: 207333058

José Daniel López Jaimes
Matricula: 207333480

Asesor: Dra. María Lizbeth Gallardo López
Profesor Investigador
Departamento de Sistemas

Enero 2013

Índice

1 Casos de Uso para la Aplicación: *SHAREDRAW*

La figura 1 muestra el diagrama de casos de uso general.

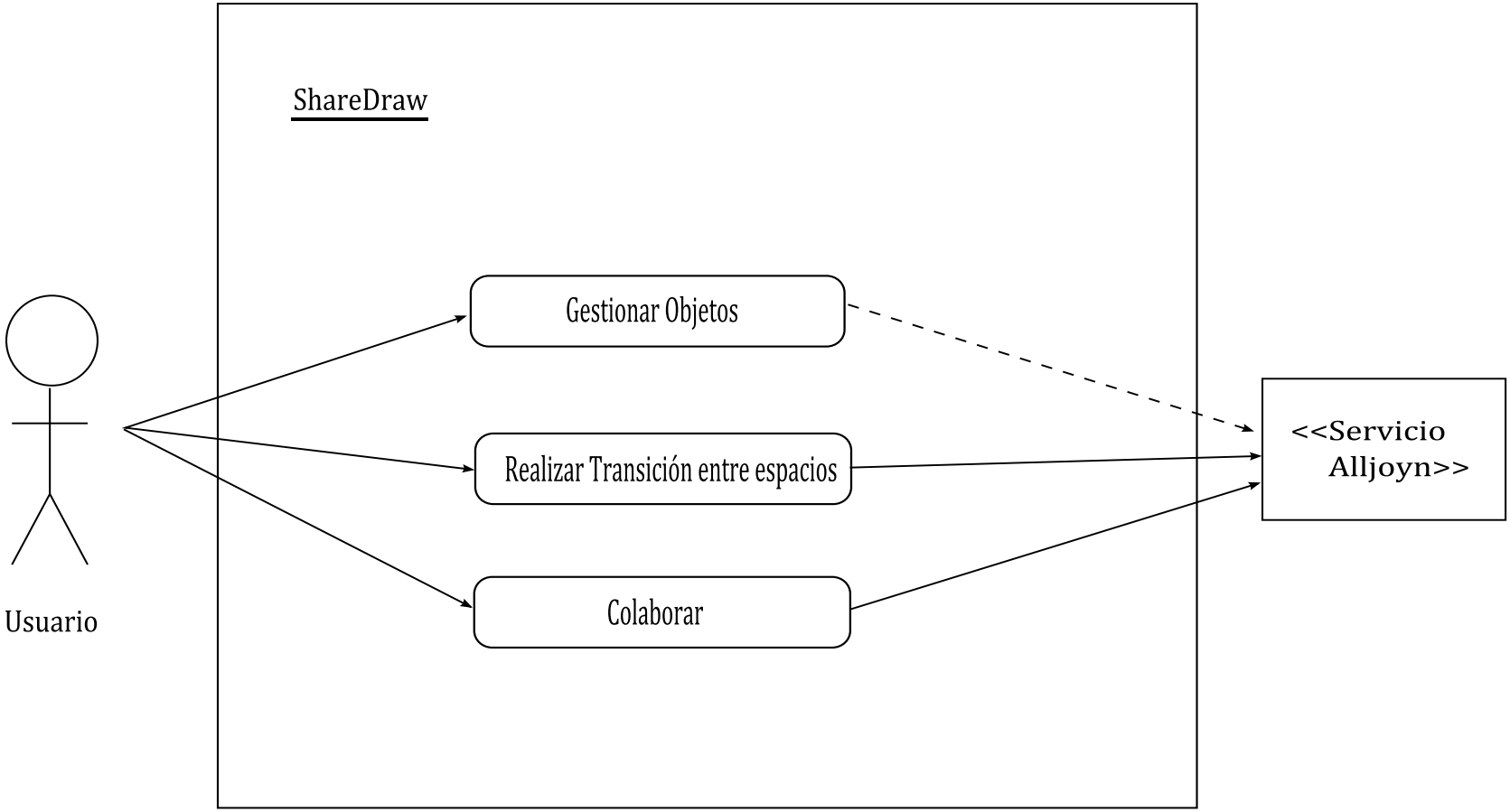


Figura 1. Diagrama de Casos de uso de ShareDraw

2 Escenarios de uso

2.1 Colaborar

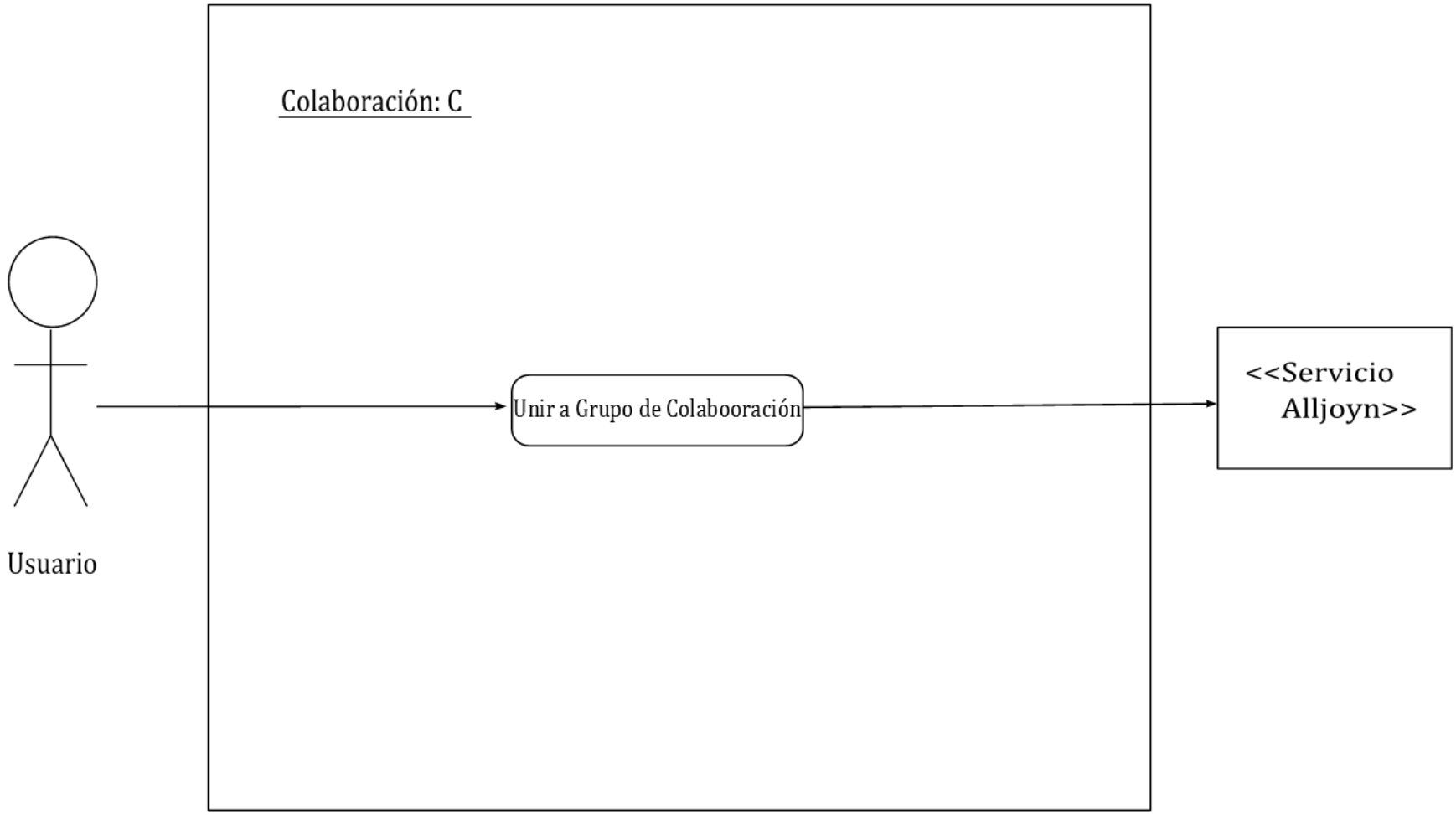


Figura 2. Diagrama de casos de uso: Colaborar.

C-1: Unir a Grupo de Colaboración

Actor primario: Usuario

Actor Secundario: Servicio Alljoyn

Disparador: El actor primario desea trabajar en modo colaborativo con otros actores primarios y un grupo de colaboración ya existente.

Precondición: Estar conectados los actores primarios a una misma red wifi. Tener instalado y activado Alljoyn. Otro actor primario ya creó un grupo de colaboración.

Poscondición: El actor primario se unió a un grupo de colaboración. Se habilita el espacio de trabajo público para la colaboración.

Flujo Principal:

1. El actor primario selecciona la opción modo colaborativo[1].
2. La aplicación muestra la opción de Unirse a un grupo de Colaboración.
3. El actor primario selecciona la opción de Unirse a un Grupo de Colaboración.
4. La aplicación mostrará un mensaje de confirmación y se habilitará el espacio público.

Flujo Alternativo:

5. El actor primario de un toque en cancelar.

5.1. El actor primario no se unirá a un grupo de colaboración. La aplicación retorna a un estado anterior al de seleccionar la opción modo colaborativo.

Requerimientos no funcionales:

En caso de que exista algún fallo se mostrará un mensaje al actor primario.

En caso de que la aplicación tardara realizar la acción, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento.

[1] Boton principal al inicio de la aplicación o Botón con Figura de Android y un signo más.

2.2 Gestionar Objetos

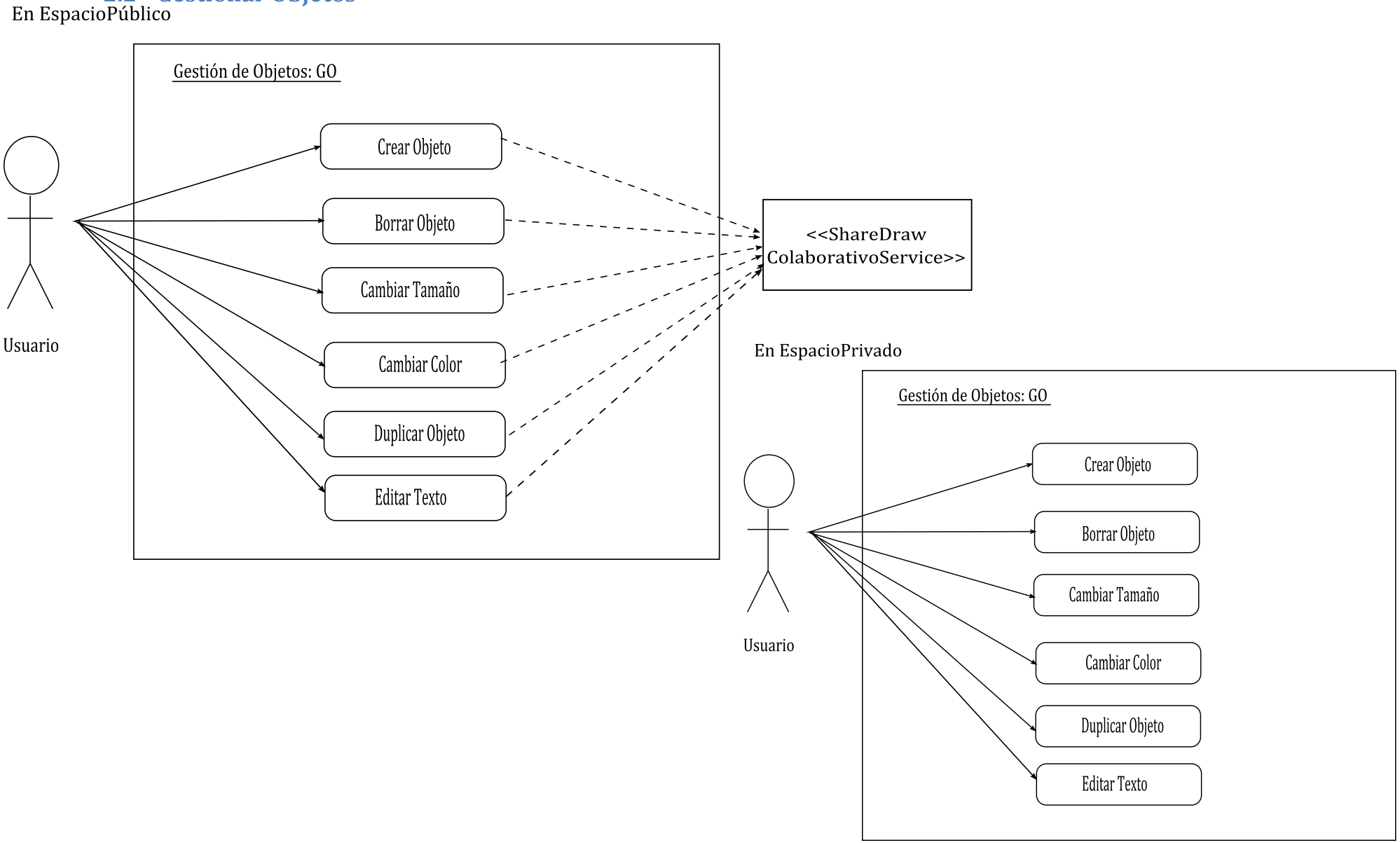


Figura 3.Casos de uso ShareDraw: GestionarObjetos

GO-1: *Crear Objeto*

Actor Primario: Usuario

Actor Secundario: -

Disparador: El actor primario desea crear un **objeto**, ya sea una **figura** o **texto**, en el espacio de trabajo donde se encuentre.

Precondición: El actor primario se encuentra en un espacio de trabajo (privado o público).

Poscondición: La aplicación crea el objeto y lo muestra en el espacio.

Flujo Principal:

1. El actor primario oprime tecla o ícono menú, que contiene cualquier dispositivo con Android.
2. El actor primario selecciona, en el menú principal, la opción de Crear Figura[2].
3. La aplicación muestra un diálogo con las siguientes figuras a elegir: **línea, círculo**, **triángulo**, **cuadrado** o **pentágono** o signo más(polígonos), la cual al ser seleccionada mostrará una lista con los número de **lados** posibles a elegir para la figura poligonal (5, 6, 7, 8).
4. El actor primario selecciona la figura o número de lado deseado.
5. El diálogo desaparece y aparecerá un mensaje "Toca la pantalla".
6. El usuario tocará la pantalla. La figura se creará y mostrará en la posición tocada por el usuario y en el **espacio** donde se encuentre el actor primario.

Flujo Alternativo:

1. El actor primario selecciona la opción de crear texto[3].

   1.1 La aplicación mostrará un diálogo con un cuadro de texto sobre el espacio de trabajo, de igual forma se mostrará un teclado virtual.

   1.2 El actor primario introducirá el texto deseado y deberá seleccionar "enter" al finalizar el texto.

   1.3 El teclado virtual se esconderá y aparecerá un mensaje "Toca la pantalla".

[2] Botón con tres figuras geométricas.
[3] Botón con letra A.

1.4 El actor primario tocará la pantalla. El **texto** escrito se mostrará en la posición tocada por es usuario y en el espacio donde se encuentre el actor primario.

Requerimientos no funcionales:

En caso que el usuario se encuentre en el espacio público gestionando un objeto, será necesario comunicarse con el servicio de Alljoyn, para la actualización de los cambios realizados.

En caso de que la aplicación tardara en mostrar el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento.

GO-2: *Borrar Objeto*

Actor primario: Usuario

Actor Secundario:-

Disparador: El actor primario desea borrar una figura o texto de alguno de sus espacios.

Precondición: Se tiene al menos un objeto (figura o texto) que borrar en algún espacio de trabajo.

Poscondición: El objeto seleccionado se borrará del espacio.

Flujo Principal:

1. El actor primario selecciona el objeto por más de un segundo sin moverlo de lugar.
2. El objeto mostrará una **Menú Objeto** .
3. El actor primario seleccionará la **opción** de borrar[4].
4. La aplicación mostrará un diálogo de confirmación.
5. El objeto se eliminará del espacio donde se encontraba. En caso de que se encontrará en el espacio público el objeto, de igual forma se eliminará de los espacios públicos de los demás usuarios.

Flujo Alternativo:

1. El actor primario selecciona al objeto por más de un segundo, en movimiento.

[4] Botón con bote de basura.

1.1. El objeto pasará a un estado de enfoque para cambiar tamaño GO-3.

Requerimientos no funcionales:
En caso que el usuario se encuentre en el espacio público gestionando un objeto, será necesario comunicarse con el servicio de Alljoyn, para la actualización de los cambios realizados.

En caso de que la aplicación tardara en borrar el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento.

En caso de que exista algún fallo se mostrará un mensaje al actor primario.

GO-3: *Cambiar Tamaño*

Actor Primario: Usuario

Actor secundario:-

Disparador: El actor primario desea cambiar el tamaño de algún objeto.

Precondición: Se tiene al menos un objeto (figura o texto) en algún espacio de trabajo.

Poscondición: El objeto seleccionado habrá cambiado de tamaño.

Flujo Principal:
1. El actor primario selecciona el objeto mediante el gesto multitáctil ".
2. El objeto se encuentra enfocado.
3. El actor primario, mediante el gesto de "pellizco", podrá agrandar el objeto posicionando dos dedos sobre éste y separarlos sin despegarlos de la pantalla.
4. El objeto cambiará de tamaño[5], ya sea en el **espacio privado** o público del actor primario. En caso de que se encontrará en el espacio público el objeto, de igual forma cambiará de tamaño en los espacios públicos de los demás **usuarios**.

Flujo Alternativo:

[5] El tamaño del objeto variará según el tamaño del "pellizco".

3. El actor primario, mediante el gesto de "pellizco", podrá reducir el tamaño del objeto posicionando dos dedos sobre éste y juntarlos sin despegarlos de la pantalla. Se regresa el flujo al punto 4.

Requerimientos no funcionales:

En caso que el usuario se encuentre en el espacio público gestionando un objeto, será necesario comunicarse con el servicio de Alljoyn, para la actualización de los cambios realizados.

En caso de que la aplicación tardara en cambiar de tamaño el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento. En caso de que exista algún fallo se mostrará un mensaje al actor primario.

GO-4: *Cambiar Color*

Actor primario: Usuario

Actor Secundario:-

Disparador: El actor primario desea cambiar de color algún objeto de su espacio de trabajo, ya sea público o privado

Precondición: Se tiene al menos un objeto (figura o texto) en algún espacio de trabajo.

Poscondición: El objeto seleccionado habrá cambiado de color.

Flujo Principal:

1. El actor primario selecciona el objeto por más de un segundo sin moverlo de lugar.
2. El objeto mostrará un Menú Objeto.
3. El actor primario seleccionará la opción de color[6].
4. La aplicación muestra una **paleta de colores** .
5. El actor primario coloca su dedo sobre la paleta de colores recorriéndolo por todos los colores que desee. Da un toque[7] en el centro del seleccionador de color para confirmar.
6. El objeto cambiará de color en cualquier espacio en el que se encuentre.

[6] Botón con paleta de nueve colores.
[7] Toque se refiere a un gesto rápido para seleccionar un botón o en este caso un color.

Flujo Alternativo:

5. El actor coloca su dedo sobre la paleta de colores. No elige ningún color de la paleta. Se sale de la paleta de colores dando un toque en el botón de retorno de Android.

   5.1. La aplicación esconde la paleta de colores. El objeto seleccionado con anterioridad no presenta cambio en el color.

Requerimientos no funcionales:

En caso que el usuario se encuentre en el espacio público gestionando un objeto, será necesario comunicarse con el servicio de Alljoyn, para la actualización de los cambios realizados.

En caso de que la aplicación tardara en cambiar de color el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento. En caso de que exista algún fallo se mostrará un mensaje al actor primario.

GO-5: *Duplicar Objeto*

Actor primario: Usuario

Actor Secundario:-

Disparador: El actor primario desea duplicar algún objeto de su espacio de trabajo, ya sea público o privado

Precondición: Se tiene al menos un objeto (figura o texto) en algún espacio de trabajo.

Poscondición: El objeto seleccionado habrá sido duplicado.

Flujo Principal:

1. El actor primario selecciona el objeto por más de un segundo sin moverlo de lugar.
2. El objeto mostrará una Menú Objeto.
3. El actor primario seleccionará la opción de duplicar[8].
4. La aplicación mostrará en el espacio de trabajo el objeto duplicado.

Requerimientos no funcionales:

[8] Botón con dos figuras iguales.

En caso que el usuario se encuentre en el espacio público gestionando un objeto, será necesario comunicarse con el servicio de Alljoyn, para la actualización de los cambios realizados.

En caso de que la aplicación tardara en duplicar el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento.

En caso de que exista algún fallo se mostrará un mensaje al actor primario.

GO-6: *Editar Texto*

Actor primario: Usuario

Actor Secundario:-

Disparador: El actor primario desea editar el mensaje de Texto algún objeto de su espacio de trabajo, ya sea público o privado

Precondición: Se tiene al menos un objeto Texto en algún espacio de trabajo.

Poscondición: El objeto Texto seleccionado habrá cambiado en su mensaje.

Flujo Principal:

1. El actor primario selecciona el objeto Texto por más de un segundo sin moverlo de lugar.
2. El objeto mostrará un Menú Objeto.
3. El actor primario seleccionará la opción de EditarTexto[9].
4. La aplicación muestra un diálogo con una caja de texto, donde se encuentra el mensaje actual del Texto.
5. El actor primario toca la caja de texto para editar el mensaje, mediante el teclado virtual. Da Aceptar cuando termine.
6. El objeto Texto cambiará en su mensaje.

Flujo Alternativo:

6. El actor toca el botón cancelar.

6.1. La aplicación esconde el teclado virtual. El objeto seleccionado con anterioridad no presenta cambio.

Requerimientos no funcionales:

[9] Botón con una Letra A y un Lápiz.

En caso que el usuario se encuentre en el espacio público gestionando un objeto, será necesario comunicarse con el servicio de Alljoyn, para la actualización de los cambios realizados.

En caso de que la aplicación tardara en cambiar la cadena del mensaje, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento. En caso de que exista algún fallo se mostrará un mensaje al actor primario.

2.3 Transición entre espacios

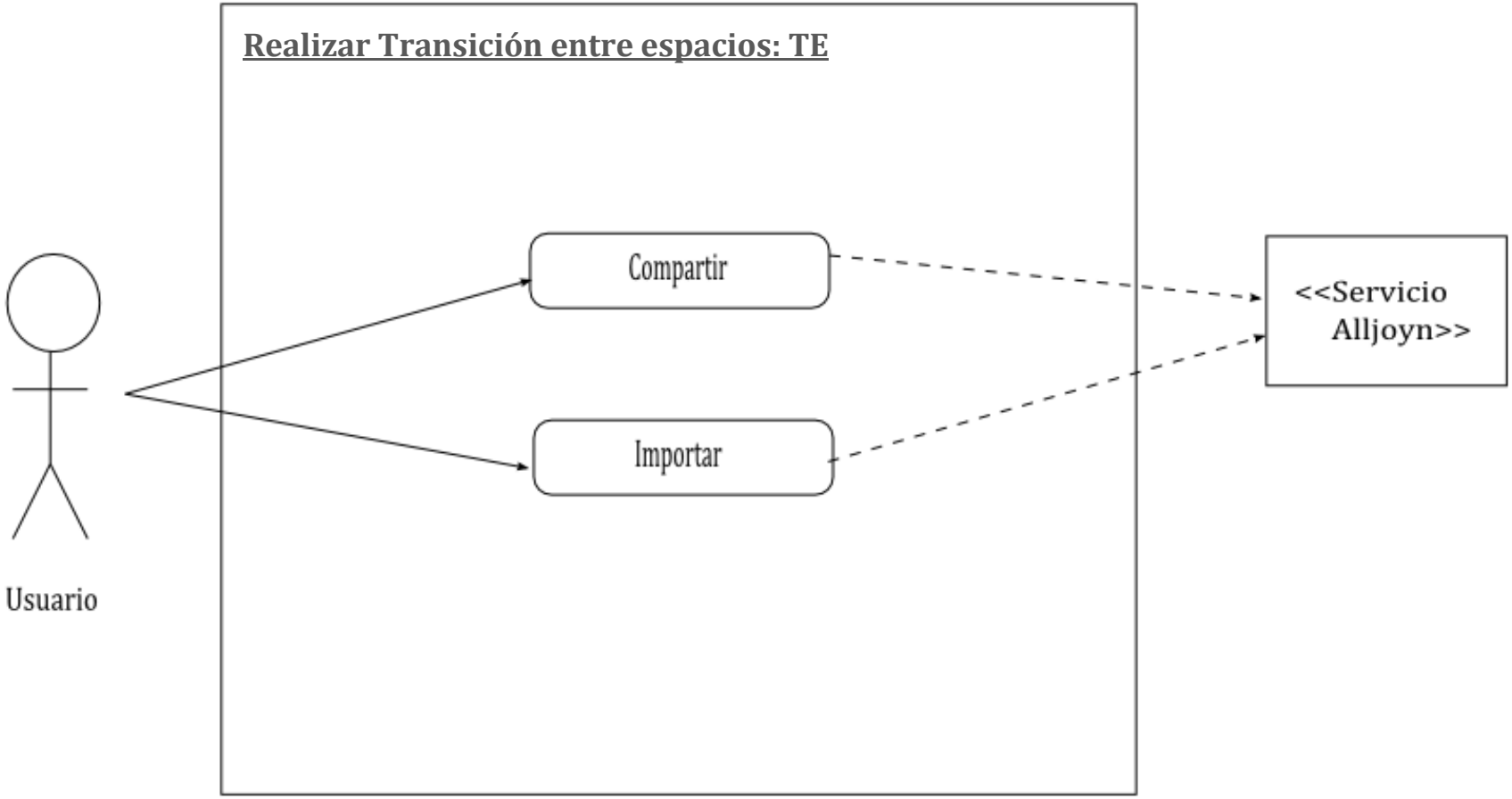


Figura4. Diagrama de Casos de uso de ShareDraw: Transición entre espacios.

TE-1*: Compartir*

Actor primario: Usuario

Actor Secundario: Servicio Alljoyn

Disparador: El actor primario desea compartir algún objeto de su espacio privado al público.

Precondición: Se tiene al menos un objeto (figura o texto) en el espacio privado. Se ha creado un grupo de colaboración[10] o se ha unido a un grupo de colaboración.

Poscondición: El objeto seleccionado será compartido al espacio público.

Flujo Principal:

1. El actor primario selecciona el objeto por más de un segundo sin moverlo de lugar.
2. El objeto mostrará una Menú Objeto.
3. El actor primario seleccionará la opción de compartir[11].
4. El actor secundario propagará el objeto a los demás actores primarios[12].
5. La aplicación emitirá un mensaje de retroalimentación confirmando que se ha compartido el objeto al espacio público. El objeto será compartido.

Requerimientos no funcionales:

En caso de que la aplicación tardara en compartir el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento.

En caso de que exista algún fallo se mostrará un mensaje al actor primario.

[10] Ver *glosario.*
[11] Botón con icono de compartir utilizado por Android.
[12] En caso de que existan más actores primarios unidos al mismo grupo de colaboración se realizará la propagación.

TE-2: *Importar*

Actor primario: Usuario

Actor Secundario: -

Disparador: El actor primario desea importar algún objeto de su espacio público al privado.

Precondición: Se tiene al menos un objeto (figura o texto) en el espacio público. Se ha creado un grupo de colaboración o se ha unido a un grupo de colaboración.

Poscondición: El objeto seleccionado será importado al espacio privado del actor primario.

Flujo Principal:

1. El actor primario selecciona el objeto por más de un segundo sin moverlo de lugar.
2. El objeto mostrará un Menú Objeto..
3. El actor primario seleccionará la opción de importar.
4. La aplicación emitirá un mensaje de retroalimentación confirmando que se ha importado el objeto al espacio privado. El objeto será importado.

Requerimientos no funcionales:

En caso de que la aplicación tardara en compartir el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento.

En caso de que exista algún fallo se mostrará un mensaje al actor primario

3 Análisis empleando Tarjetas CRC

3.1 Obtención de Entidades Claves

Entidades Candidato	Razón de eliminación	Nombre de la Entidad
Usuario		Usuario
Grupo de colaboración		Colaboración
Aplicación		
Cuadro de texto	No es perdurable	
Teclado virtual	No es perdurable	
Mensaje	No es perdurable	
Espacio	Engloba espacio público y privado	
Espacio Público		EspacioPúblico
Espacio Privado		EspacioPrivado
Lista de grupos	No es perdurable, es una colección	
Objeto Gráfico		ObjetoGráfico
Figura		PolígonoSimple
Texto		Texto
Menú Principal	No es perdurable	
Menú Objeto	No es perdurable	
Círculo	Es un atributo de figura	
Triángulo	Es un atributo de figura	
Cuadrado	Es un atributo de figura	
Figura poligonal		Polígono
Paleta de colores	No es perdurable	

3.2 Tarjetas CRC

ManejadorEspacios	
Resposabilidades: Cambio entre Espacios	Colaboración: EspacioPrivado EspacioPúblico
Atributos: Lista de ObjetoGraficos EspacioPrivado. Lista de ObjetoGraficos EspacioPrivado. EspacioPrivado EspacioPúblico	

Espacio Privado	
Resposabilidades: Crear, Borrar, Duplicar, Compartir ObjetoGrafico. Cambiar tamaño y color a ObjetoGrafico Editar mensaje a Texto.	Colaboración: ObjetoGrafico
Atributos: Lista de ObjetoGraficos. Menú Principal Menú Objeto Cambio entre Espacios	

Espacio Público	
Resposabilidades: Crear, Borrar, Duplicar, Importar ObjetoGrafico. Cambiar tamaño y color a ObjetoGrafico Editar mensaje a Texto.	Colaboración: ObjetoGrafico
Atributos: Lista de ObjetoGraficos. Menú Principal Menú Objeto Cambio entre Espacios	

Colaboración	
Resposabilidades: Métodos de Alljoyn	Colaboración: Espacio Público
Atributos: Bus de comunicación	

ObjetoGrafico

Resposabilidades:	Colaboración:
get y set onDraw	PoligonoSimple Poligono Texto
Atributos:	
PosicionXY, Tamaño, Color	

Texto

Resposabilidades:	Colaboración:
get y set. duplicar()	ObjetoGrafico
Atributos:	
PosiciónXY, tamaño, color, cadena de texto.	

Poligono	
Resposabilidades: get y set. duplicar()	Colaboración: ObjetoGrafico
Atributos: PosiciónXY, tamaño, color, lados, vector de puntos.	

PoligonoSimple	
Resposabilidades: get y set. duplicar()	Colaboración: ObjetoGrafico
Atributos: PosiciónXY, tamaño, color, tipo.	

3.3 Modelo de Dominio

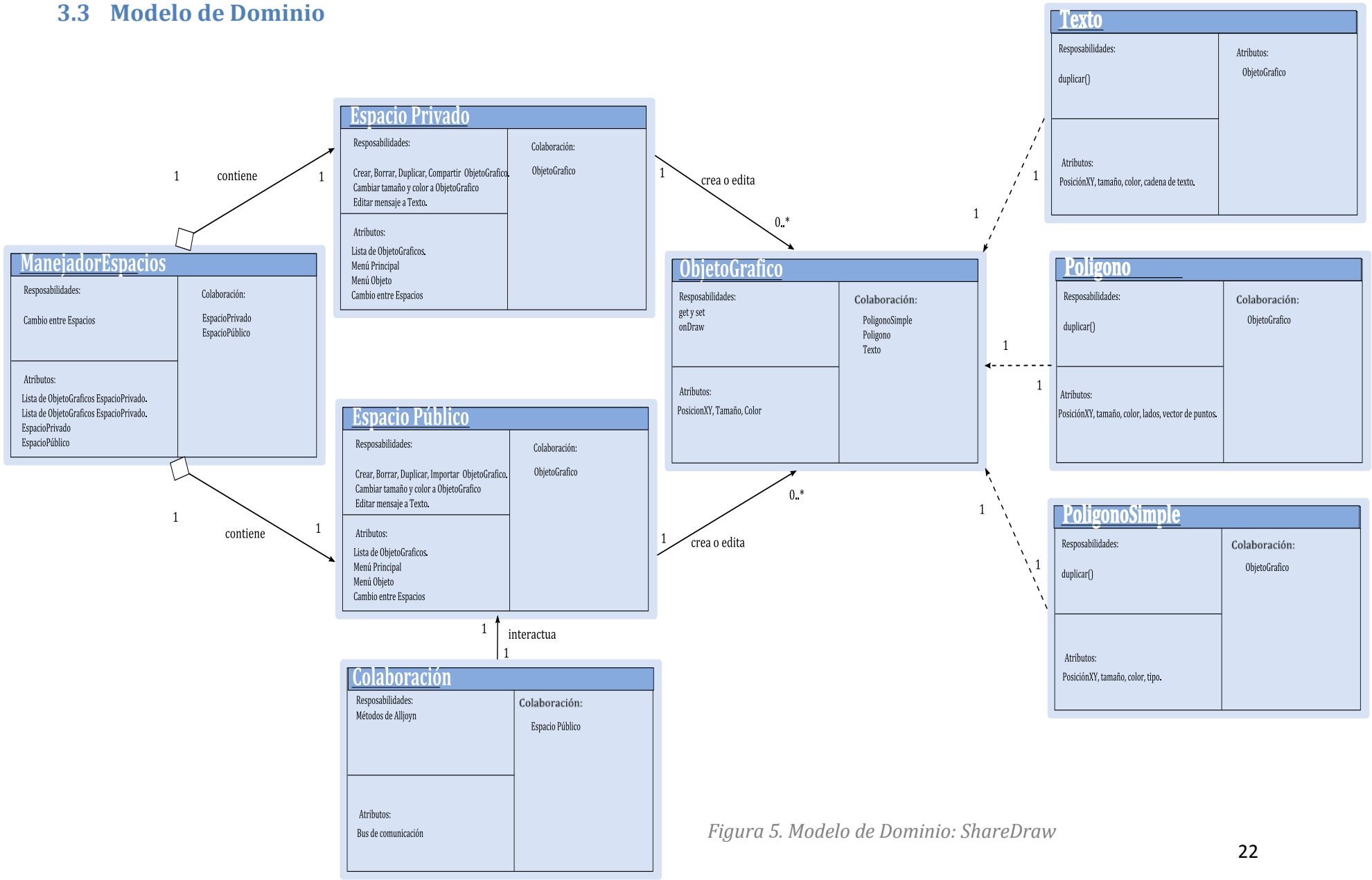


Figura 5. Modelo de Dominio: ShareDraw

4 Análisis de Robustez

4.1 Diagramas de Robustez

4.1.1 Gestión de Objetos

GO-1: Crear Objeto

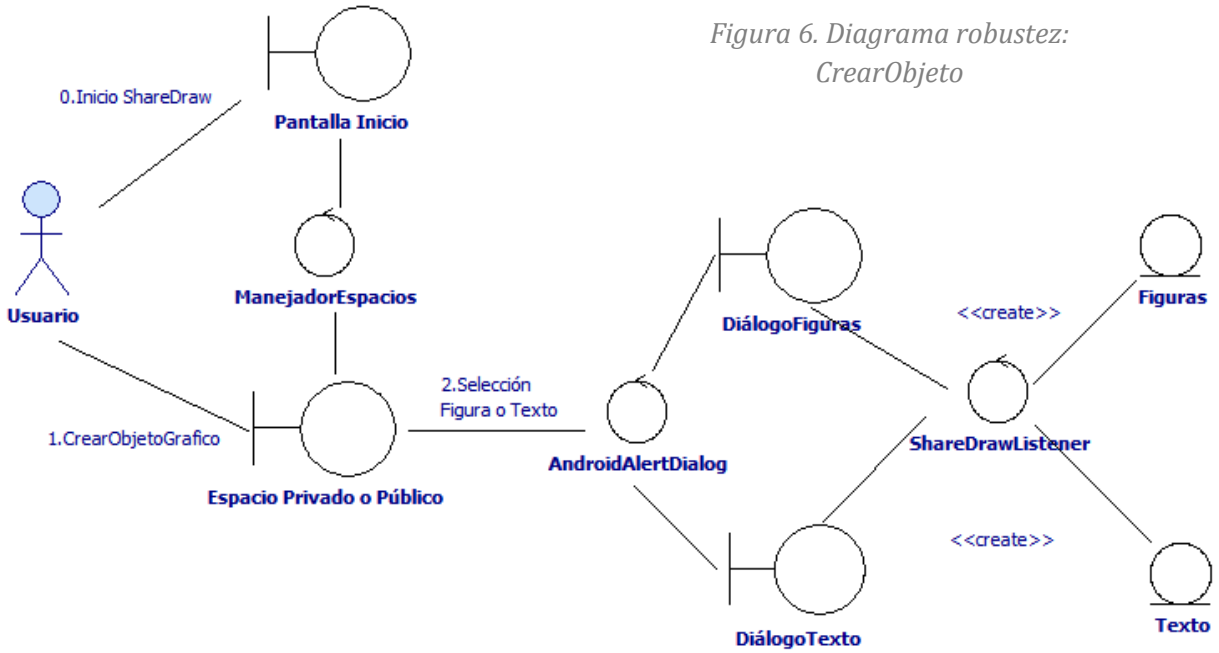


Figura 6. Diagrama robustez: CrearObjeto

GO-2: Borrar Objeto

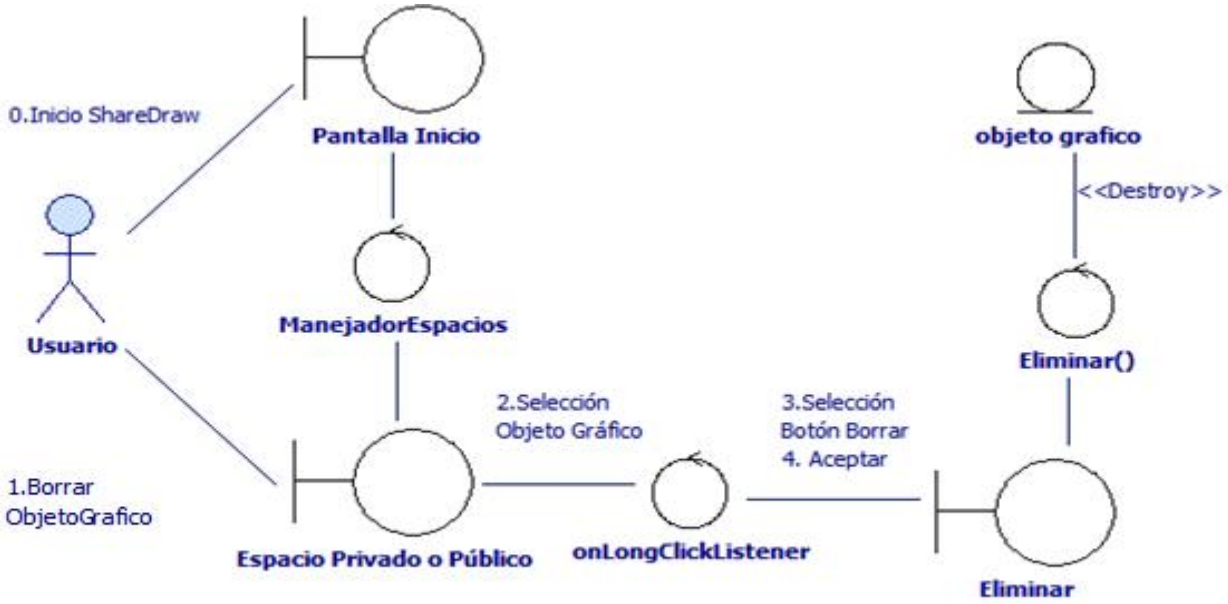


Figura 7. Diagrama robustez: Borrar Objeto

GO-3: Cambiar Tamaño

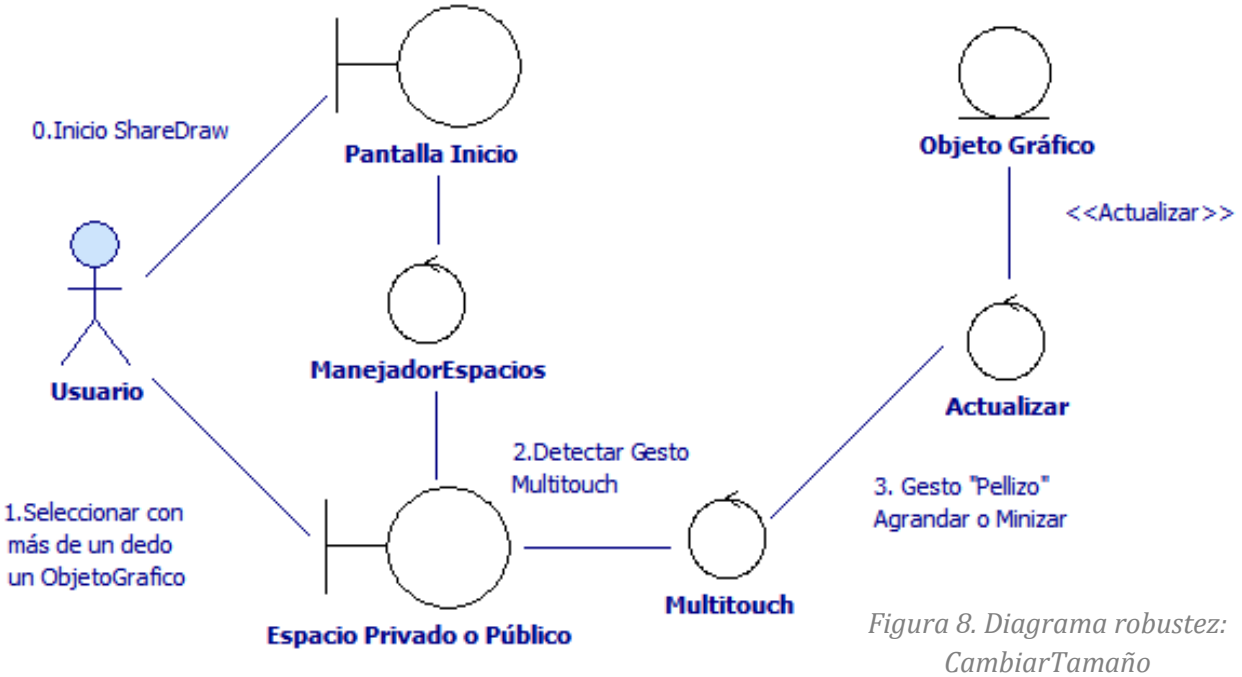


Figura 8. Diagrama robustez: CambiarTamaño

GO-4: Cambiar Color

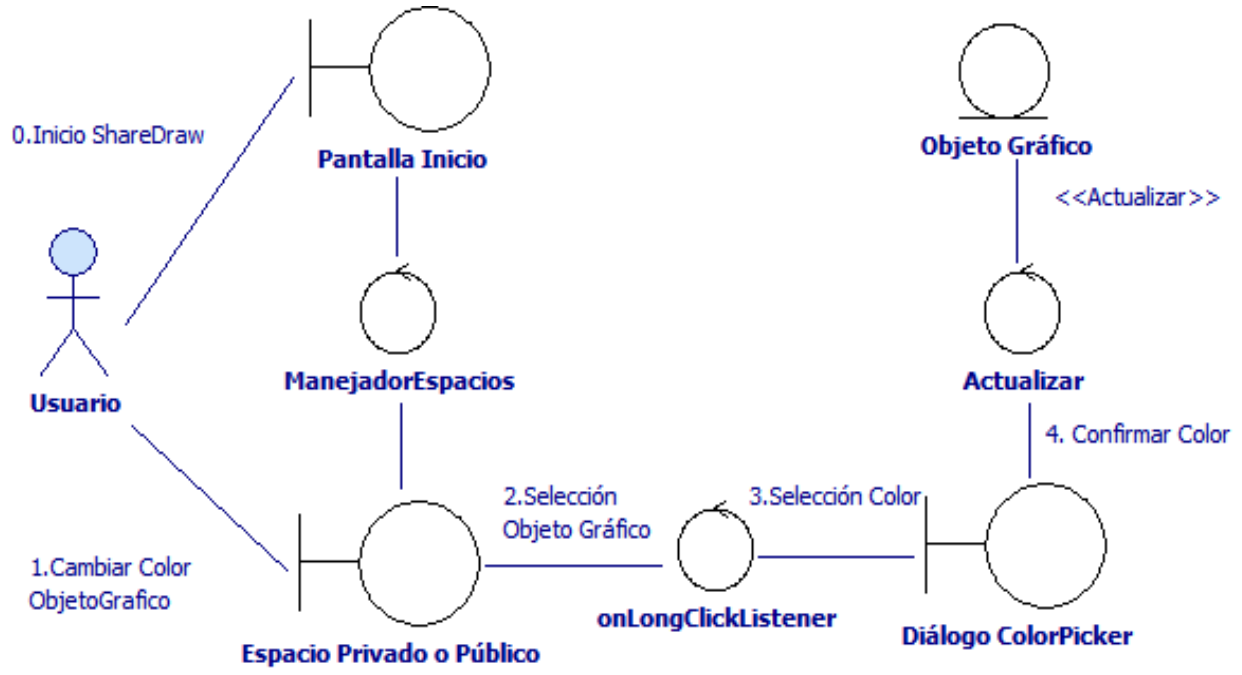


Figura 9. Diagrama robustez: Cambiar Color

GO-5: Duplicar Objeto

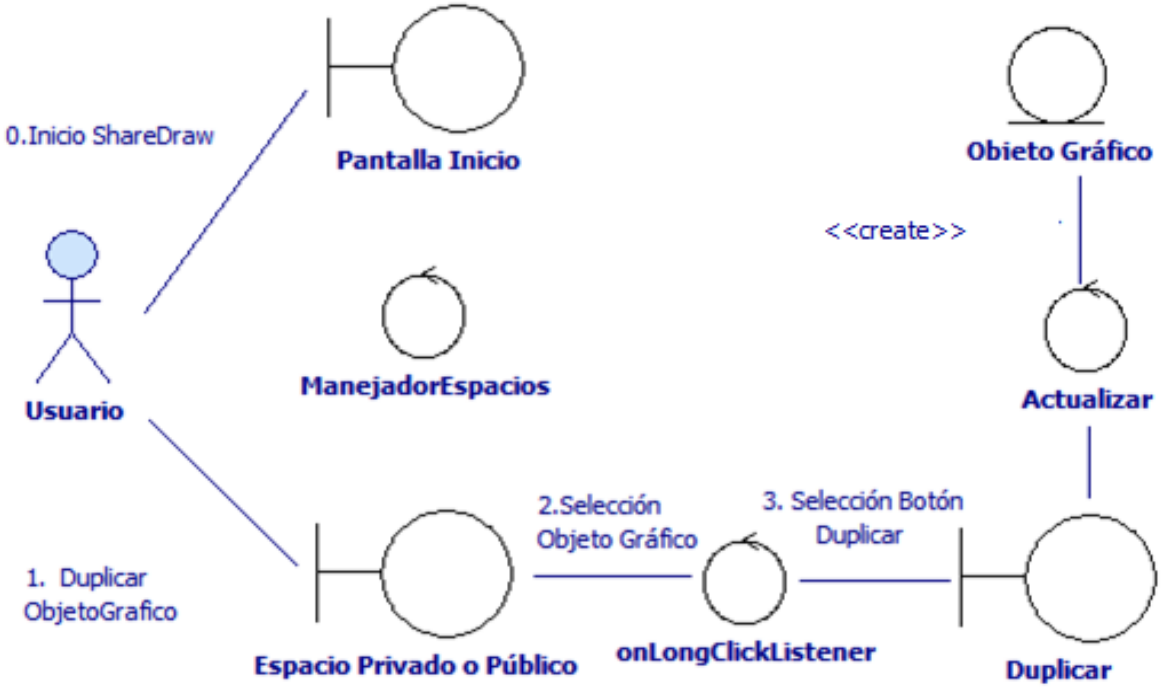


Figura 10. Diagrama robustez: Editar Texto

GO-6: Editar Texto

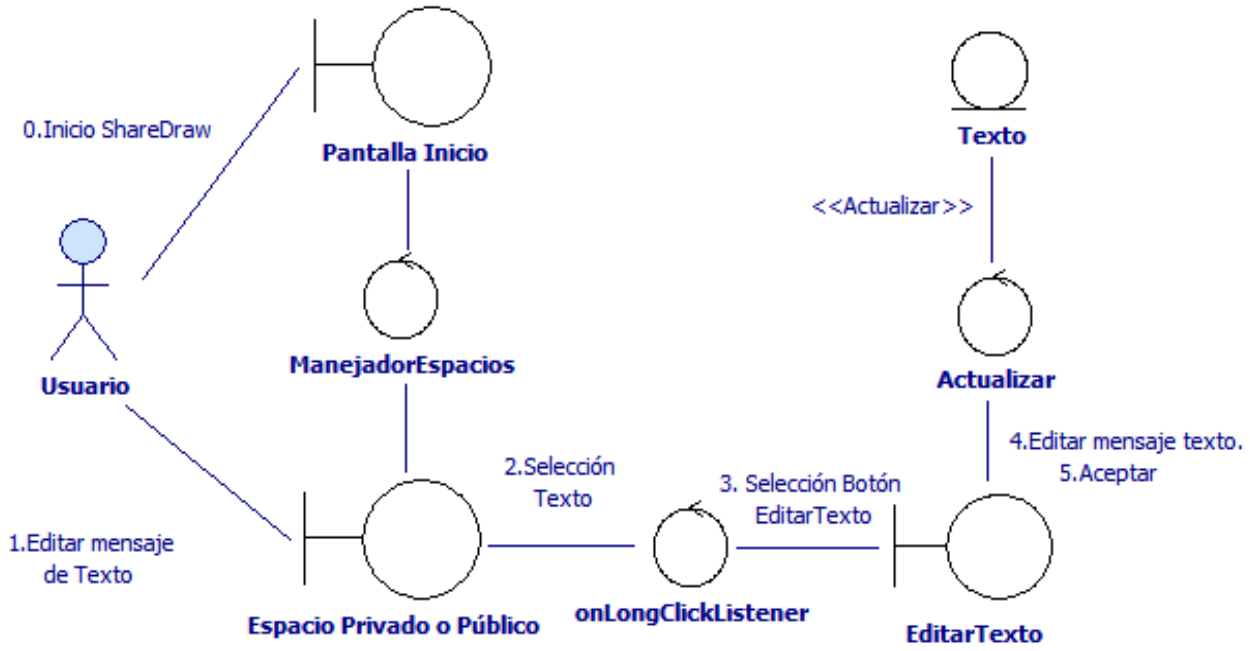


Figura 11. Diagrama robustez: Editar Texto

4.1.2 Transición entre Espacios

TE-1: Compartir

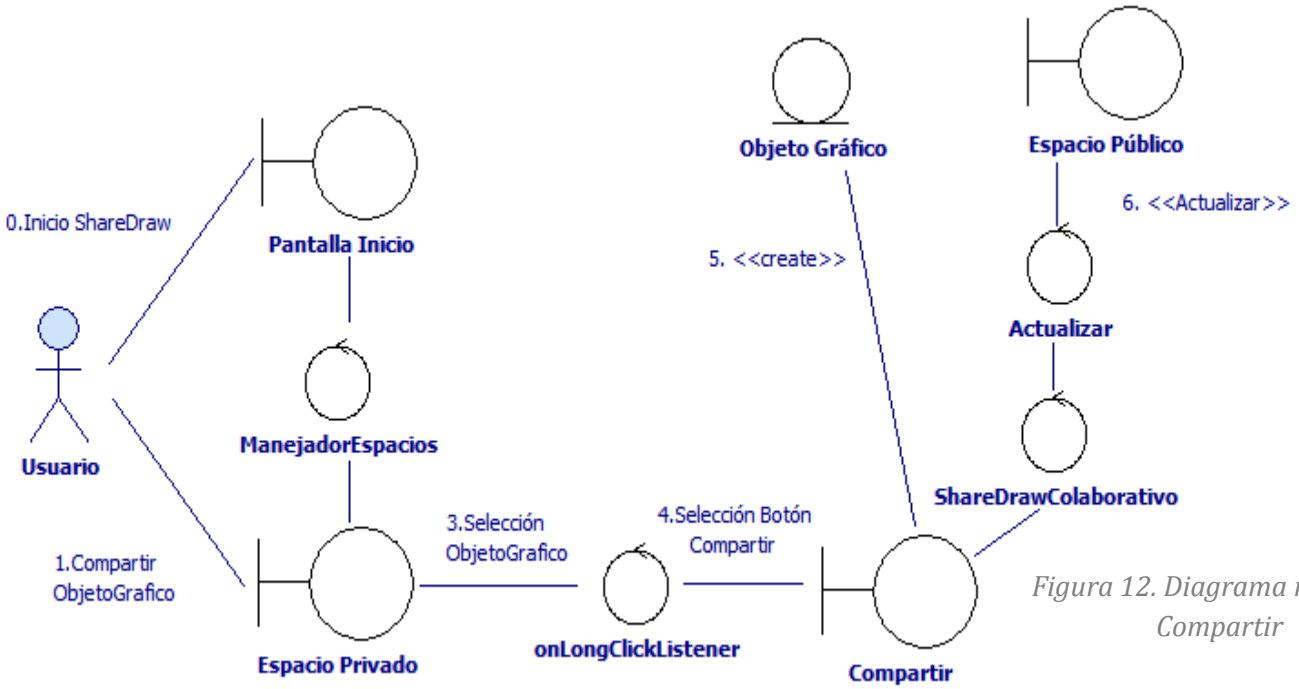


Figura 12. Diagrama robustez: Compartir

TE-2: Importar

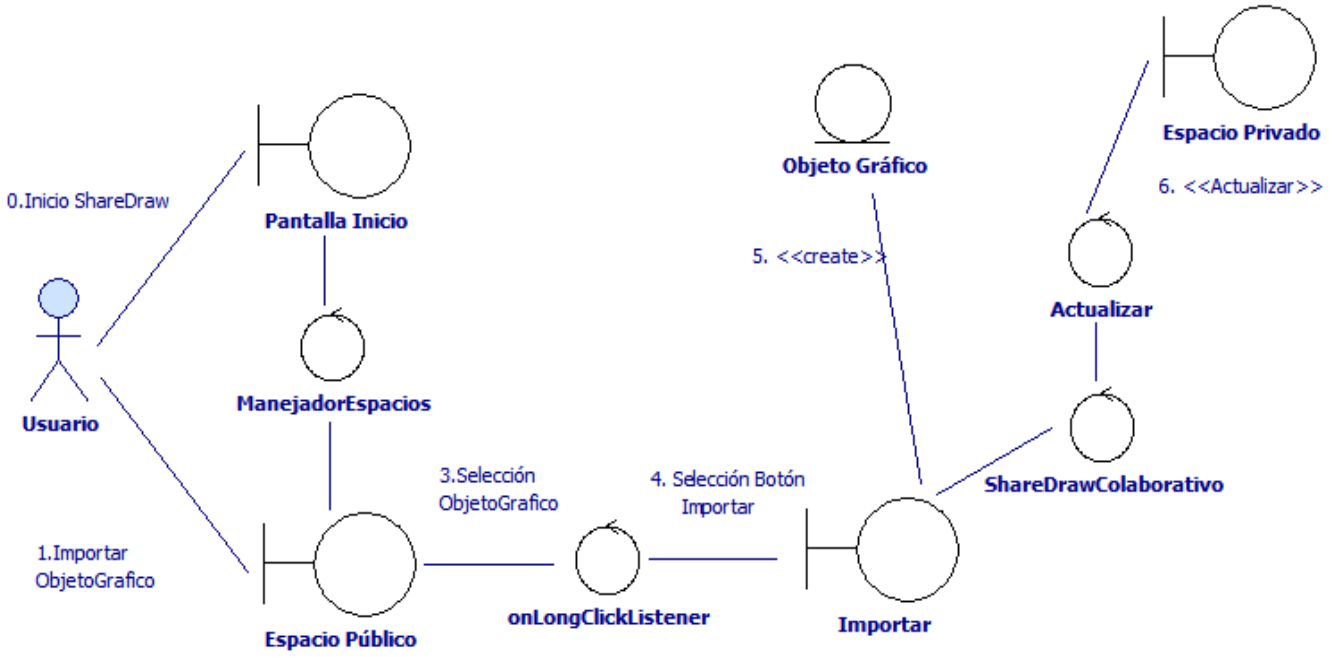


Figura 13. Diagrama robustez: Importar.

4.1.3 Colaboración

C-1: Unirse a Grupo de Colaboración

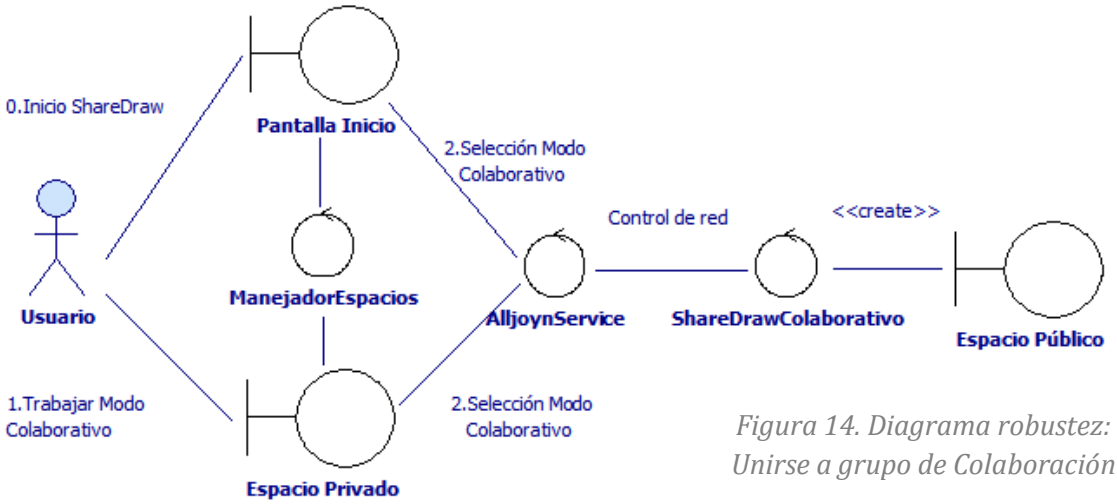


Figura 14. Diagrama robustez: Unirse a grupo de Colaboración

4.2 Diagramas de Secuencia

4.2.1 GO-1: Crear Objeto

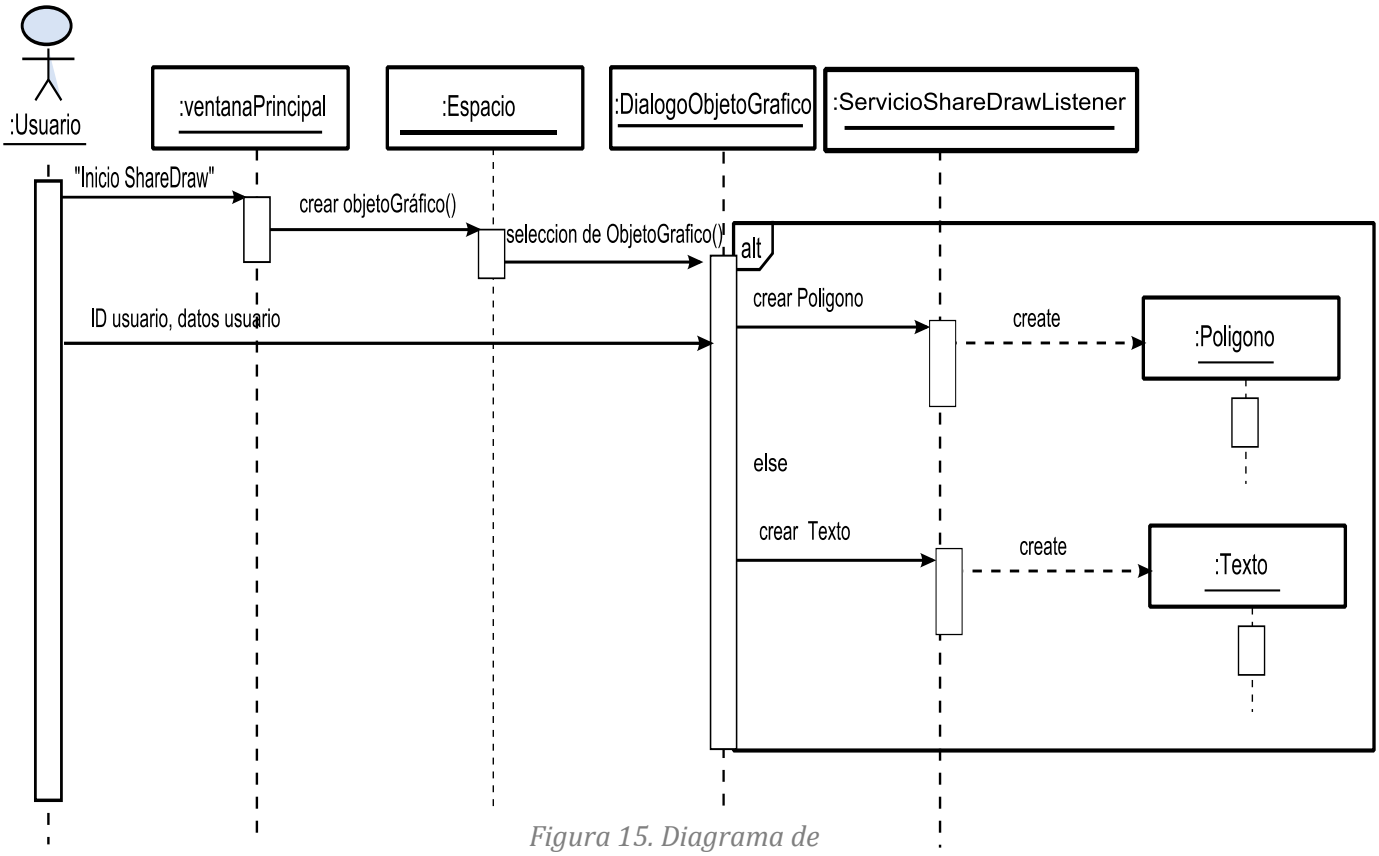


Figura 15. Diagrama de secuencia CrearObjeto

4.2.2 TE-1:Compartir

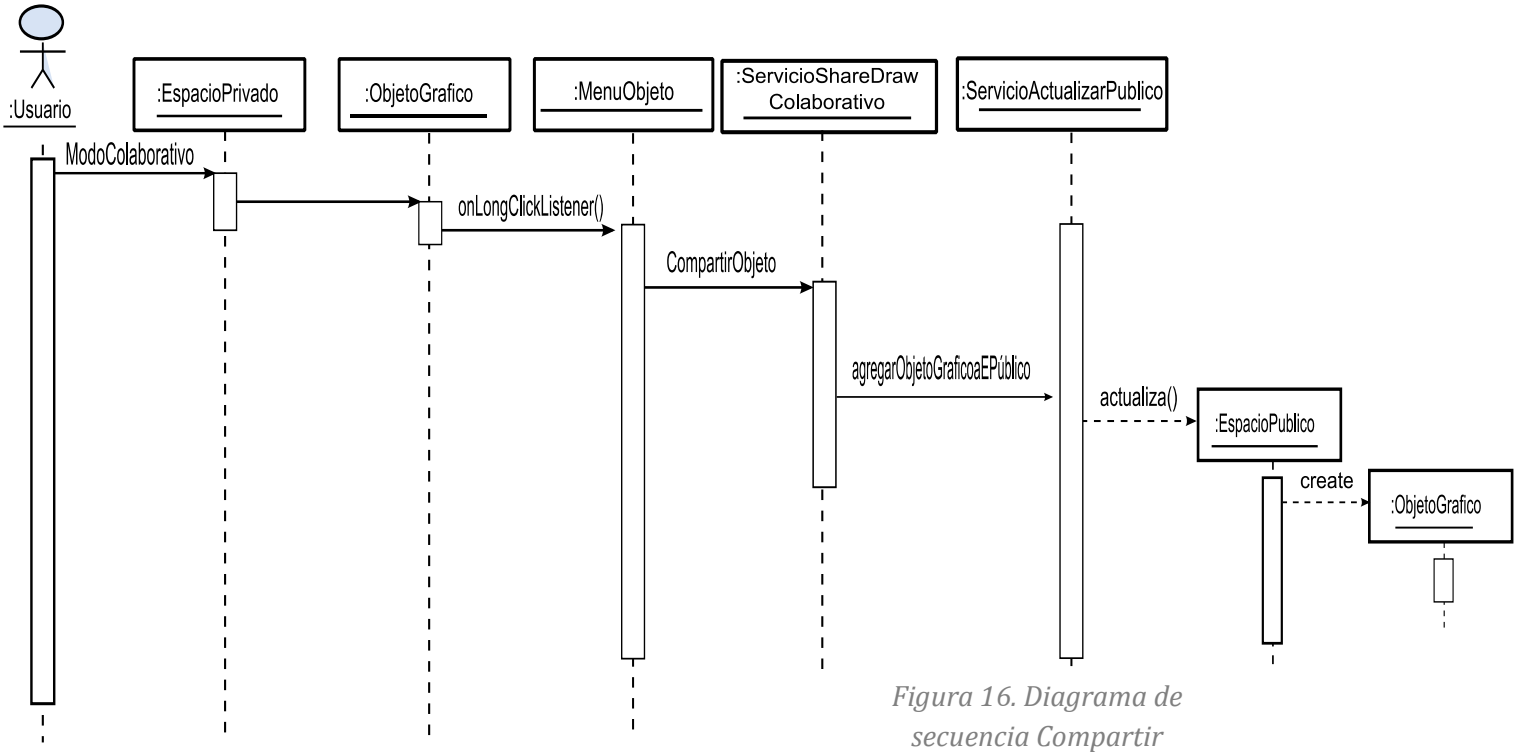


Figura 16. Diagrama de secuencia Compartir

4.2.3 C-1: Unirse a Grupo de Colaboración

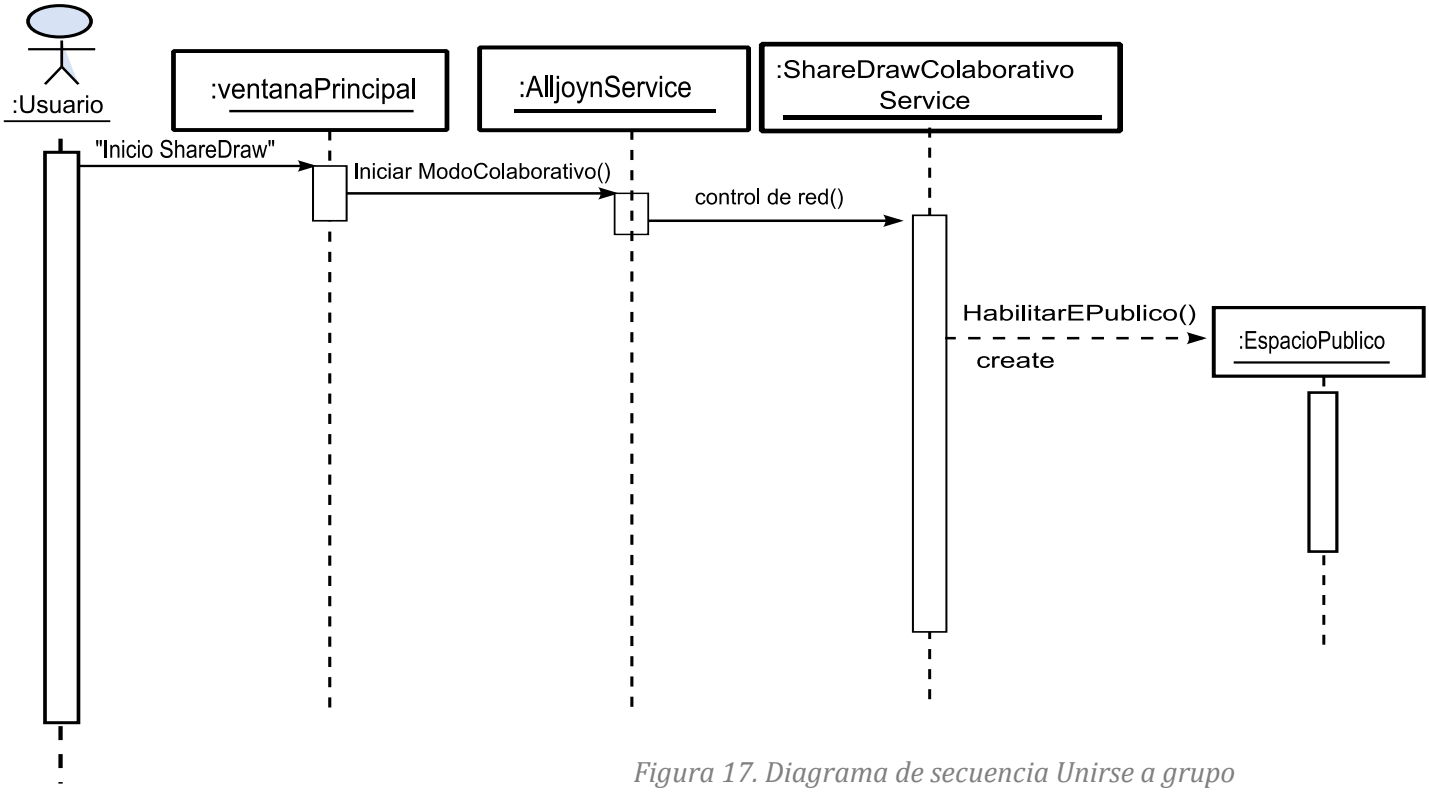
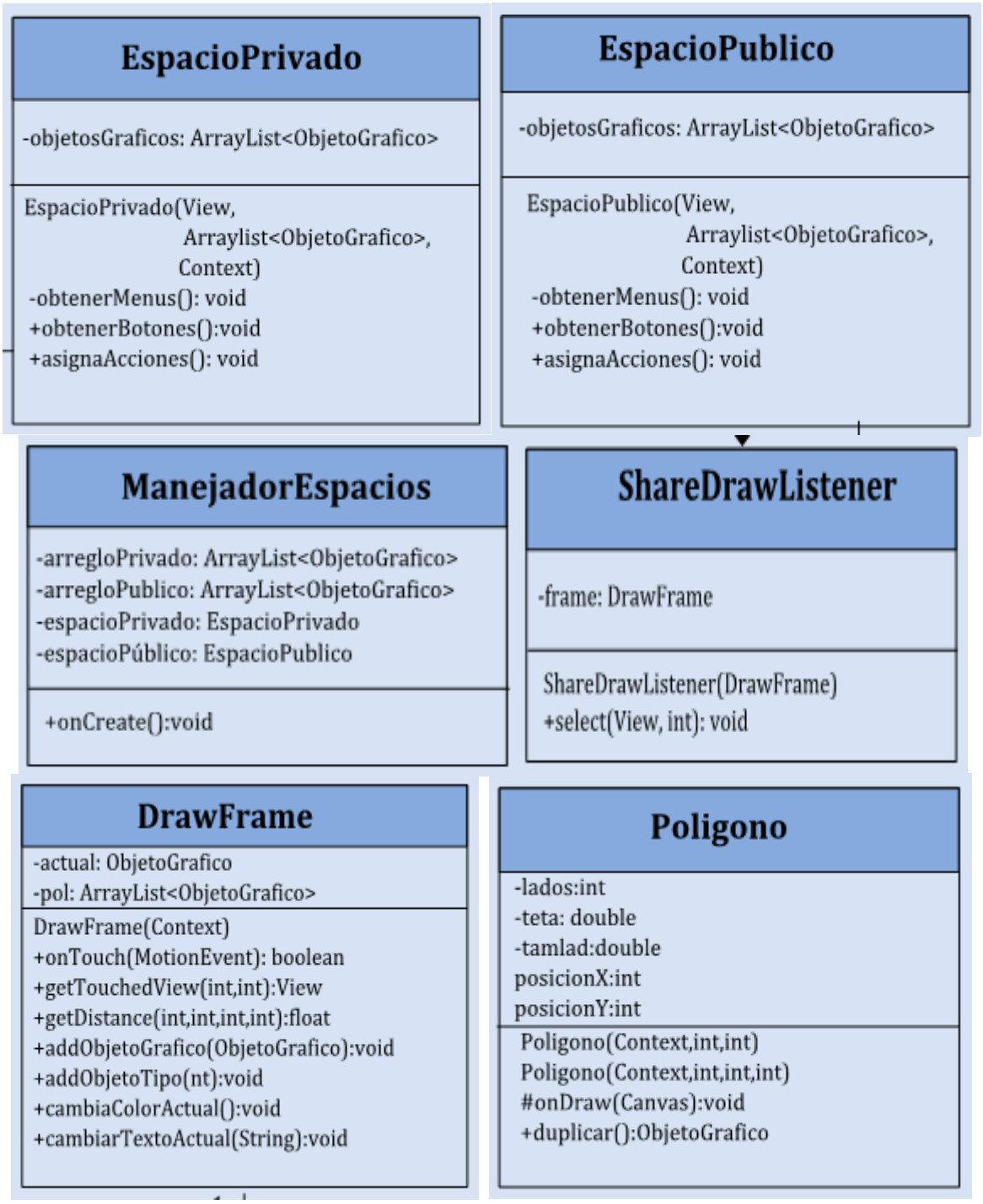


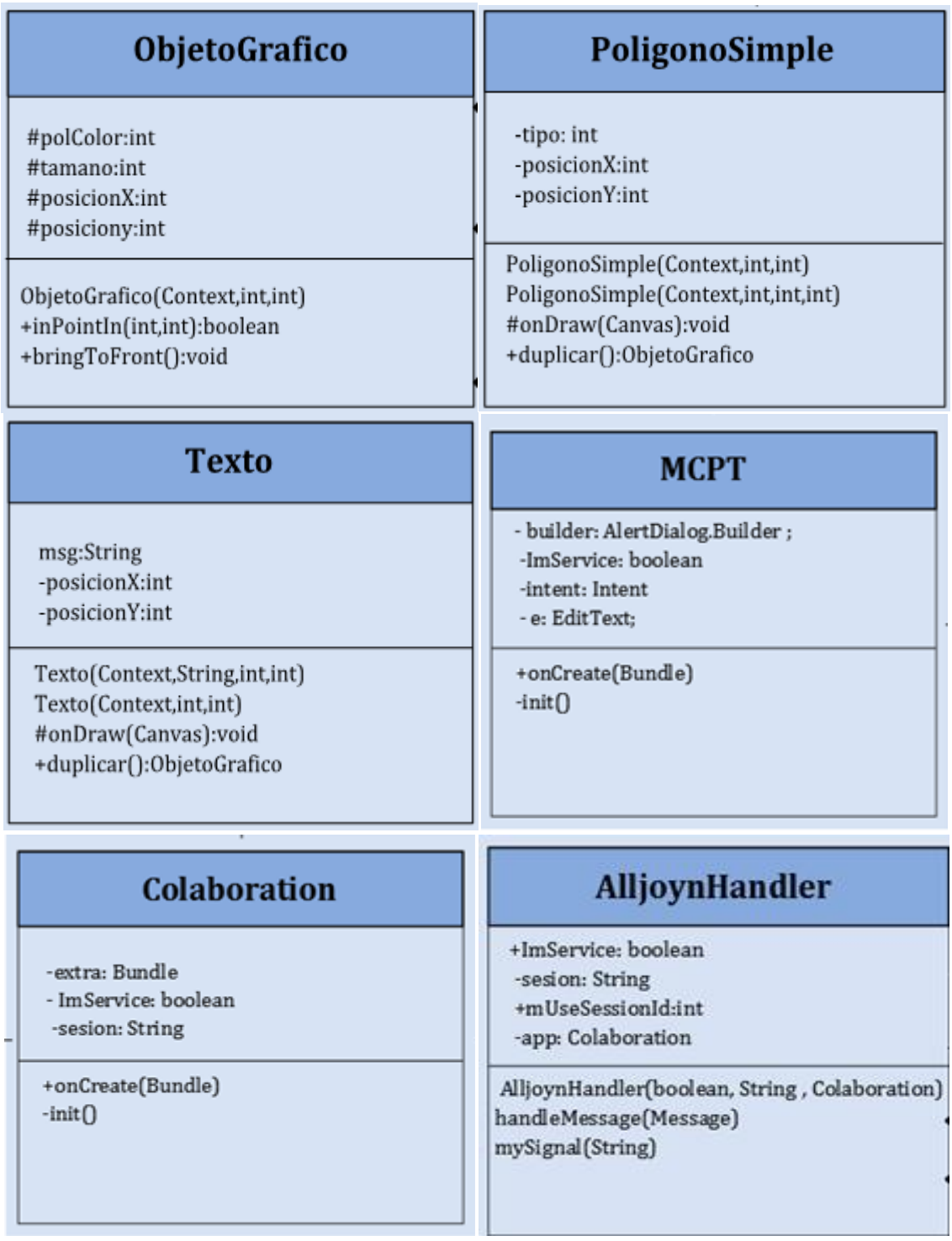
Figura 17. Diagrama de secuencia Unirse a grupo de colaboración.

5 Diagrama de Clases

5.1 Clases

Dado un análisis de los diagramas de secuencia y robustez, los casos generales de uso, el modelo de dominio y un análisis en base a la arquitectura y manejo de componentes de Android, se definieron las siguientes clases; así como su diagrama de clases:




ObjetoGrafico
#polColor:int
#tamano:int
#posicionX:int
#posiciony:int
ObjetoGrafico(Context,int,int)
+inPointIn(int,int):boolean
+bringToFront():void
PoligonoSimple
-tipo: int
-posicionX:int
-posicionY:int
PoligonoSimple(Context,int,int)
PoligonoSimple(Context,int,int,int)
#onDraw(Canvas):void
+duplicar():ObjetoGrafico
Texto
msg:String
-posicionX:int
-posicionY:int
Texto(Context,String,int,int)
Texto(Context,int,int)
#onDraw(Canvas):void
+duplicar():ObjetoGrafico
MCPT
- builder: AlertDialog.Builder ;
-ImService: boolean
-intent: Intent
- e: EditText;
+onCreate(Bundle)
-init()
Colaboration
-extra: Bundle
- ImService: boolean
-sesion: String
+onCreate(Bundle)
-init()
AlljoynHandler
+ImService: boolean
-sesion: String
+mUseSessionId:int
-app: Colaboration
AlljoynHandler(boolean, String , Colaboration)
handleMessage(Message)
mySignal(String)

5.2 Diagrama de Clases

El diagrama de clases (ver figura 18) presenta las clases del modelo lógico de la aplicación en color azul; así mismo muestra las clases de *Android* en color verde, las cuales se heredan o se implementan para el funcionamiento de *ShareDraw*.

En la figura 19 se muestra el diagrama de clases del módulo colaborativo de *ShareDraw.* Estas clases heredan o se implementan de clases de la *framework Alljoyn* (en color rojo). Tal y como abordaremos más adelante, el objetivo de utilizar las bibliotecas de *Alljoyn* es para establecer una comunicación adecuada entre dispositivos próximos en una red *p2p* sin la necesidad de recurrir a un servidor intermediario. *Alljoyn* ofrece una solución a problemas como: detección transparente de dispositivos y servicios, funcionalidades de red, el direccionamiento de mensajes. En la sección *6 Arquitectura* se presentarán las clases de *Alljoyn,* y la lógica de comunicación que se utilizará para el módulo colaborativo y aquellas interfaces o clases de *Alljoyn* necesarias.

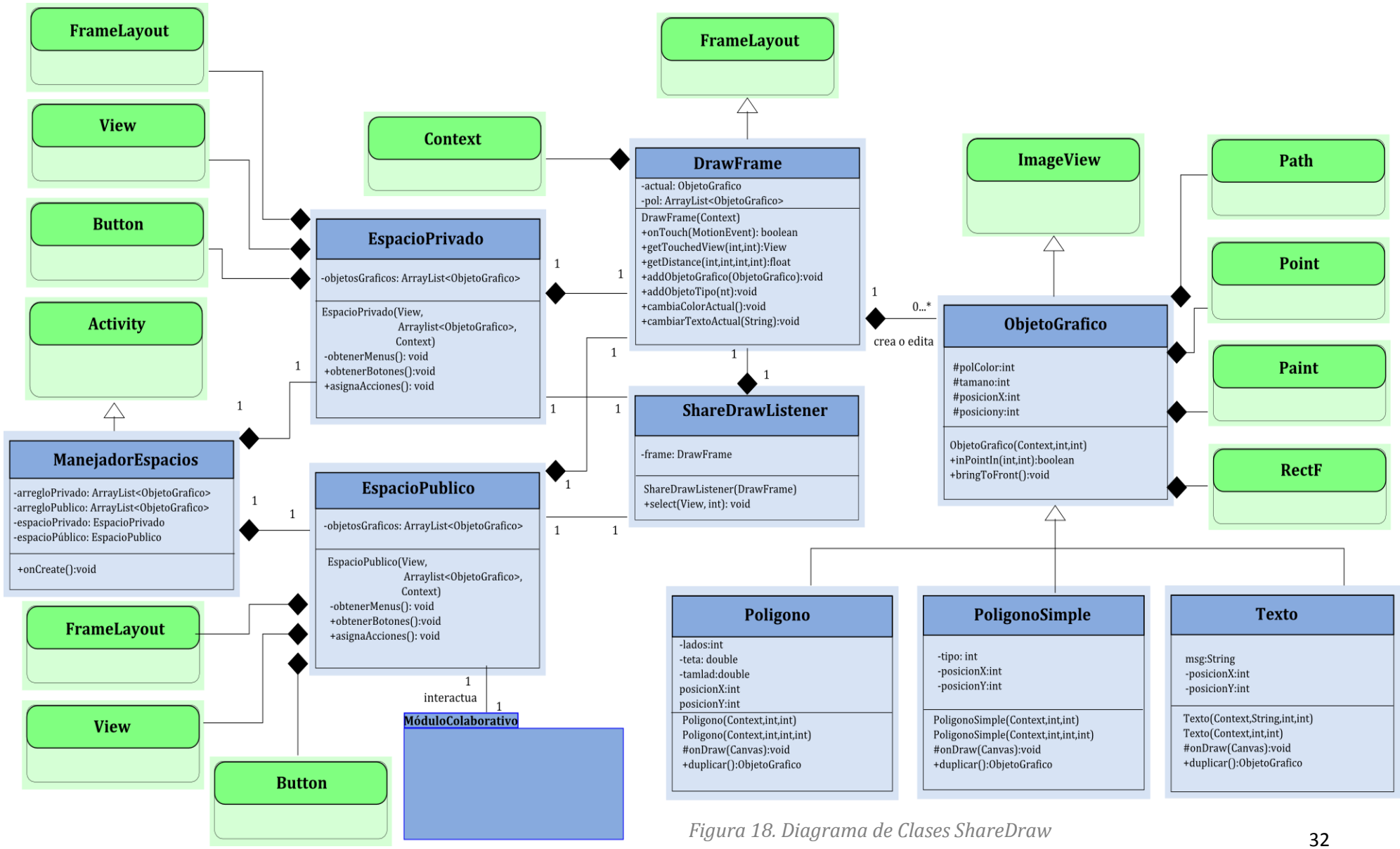


Figura 18. Diagrama de Clases ShareDraw

5.2.1 Diagrama de clases: Módulo Colaborativa

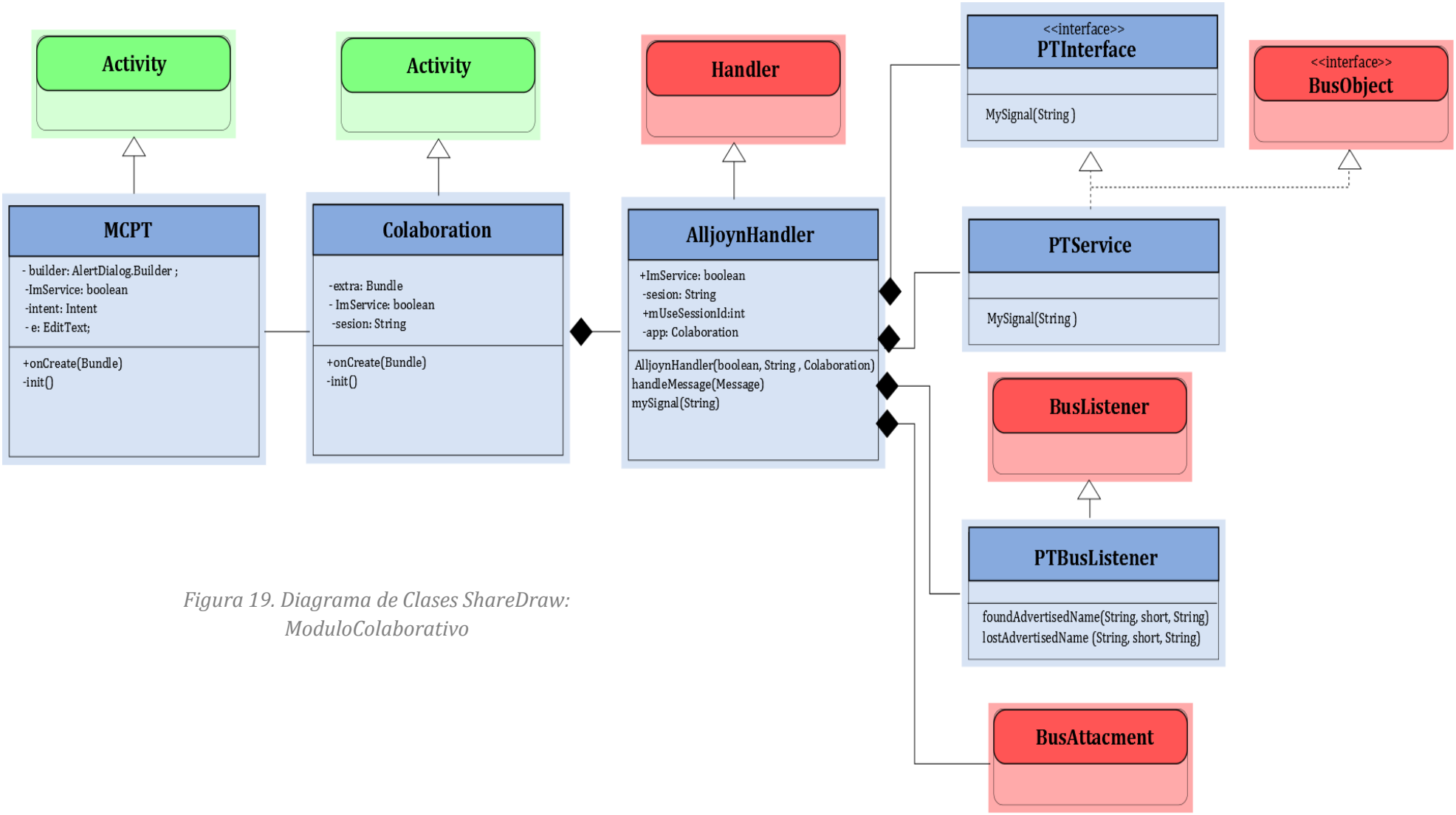


Figura 19. Diagrama de Clases ShareDraw: ModuloColaborativo

6 Arquitectura

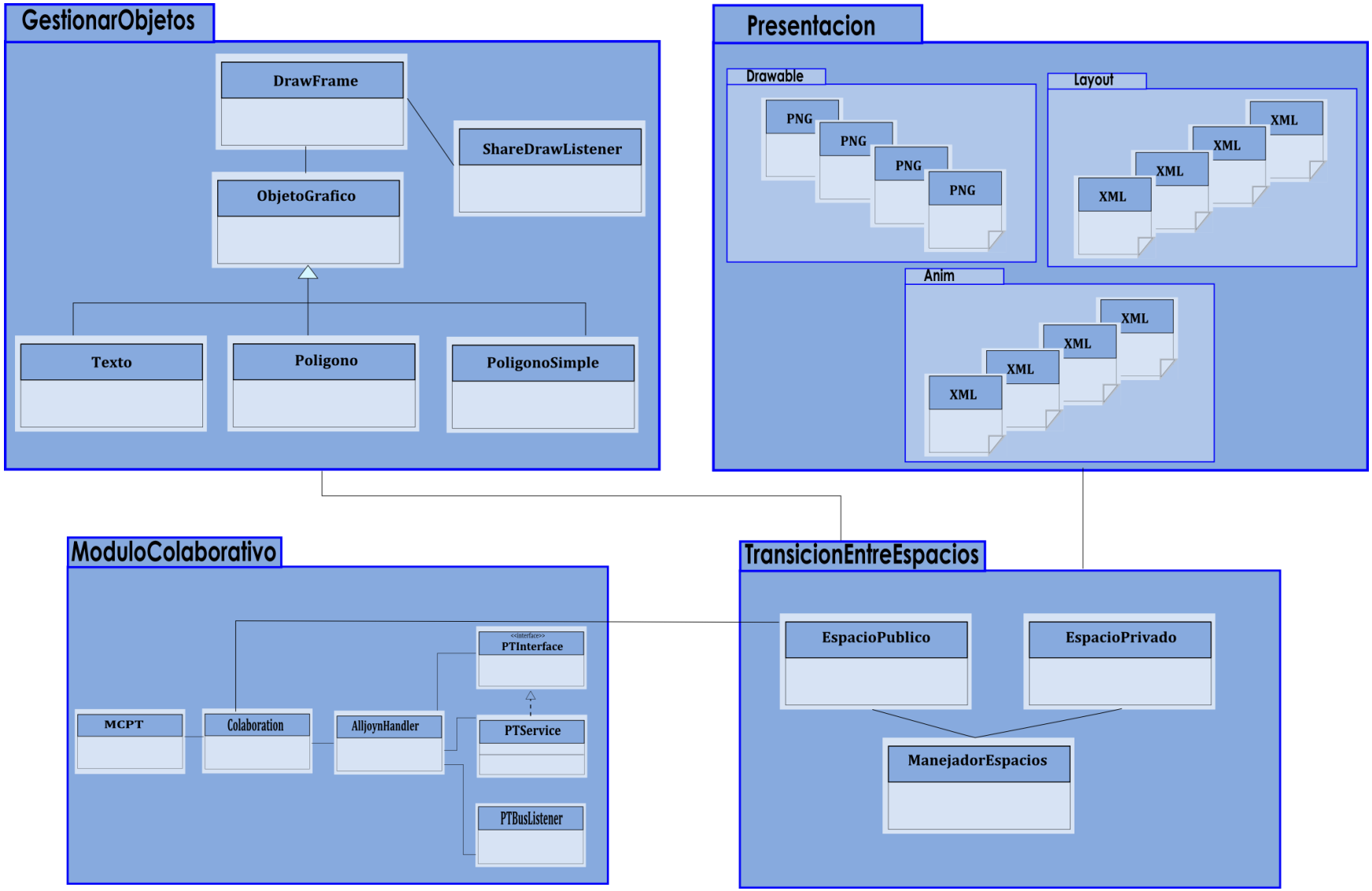


Figura 20. Arquitectura de ShareDraw

La arquitectura estará compuesta por cuatro módulos:
Capa de Gestión de Objetos. En este módulo se encuentran aquellas clases que se encargarán de la creación y edición de los objetos a dibujar en los espacios de trabajo: *Drawframe* y *ShareDrawListener.* También contiene los objetos que se gestionarán: *ObjetoGrafico, Poligono, PoligonoSimple y Texto.*

Capa de Presentación. Este módulo se compone de tres paquetes importantes dentro de una aplicación creada para *Android*: *Drawable, Layout y Anim*. El paquete *Drawable* contiene artefactos de dibujo necesarios para los iconos de la aplicación. El paquete Layout contiene los archivos *XML*, útiles para la generación de las interfaces de usuario (ventanas y diálogos) necesarias para la aplicación. El paquete Anim contiene los archivos *XML* que definen animaciones modificando las propiedades del objeto en cuestión o vistas (*View*). Para el uso de cualquiera de estos archivos o imágenes es necesario invocarlas y asignarles acciones desde las clases *EspacioPrivado* y *EspacioPúblico.*

Capa de transición entre Espacios. Este módulo contiene aquellas clases que permiten el cambio entre la vista del Espacio Privado y El Espacio Público o viceversa, así como el paso de Objetos Gráficos de un Espacio a otro.

Capa de *ModuloColaborativoShareDraw.* Las clases que componen esta capa son: *PTInterface, Alljoynhandler, MCPT, Colaboration, PTService* y *PTBusListener.* Esta capa lleva a cabo el manejo del bus de comunicación, así como de las señales de alerta de cambios en la aplicación. De igual forma se encarga de la actualización de los datos y la interacción de los objetos entre los distintos espacios públicos, siendo la clase más importante *AlljoynHandler*.

Para esta capa se hizo uso del *framework Alljoyn*, se diseñó un modelo de comunicación (ver figura 21) que permitiera definir la colaboración entre los componentes del *framework* para el módulo colaborativo. Este modelo se diseñó en base a una red Cliente Servidor para comprender los conceptos básicos de *Alljoyn*. Posteriormente, se implementó este modelo en cada dispositivo móvil obteniendo una red *peer-to-peer;* donde, cada dispositivo es, a su vez, Cliente y Servidor en este módulo de colaboración. Los componentes básicos que se requieren para implementar este modelo, son:

- ***Alljoyn Bus***: La abstracción más básica de un sistema con *AllJoyn* es *Alljoyn Bus*, el cual provee un canal de comunicación de alta velocidad que transporta mensajes a través del sistema distribuido. La ventaja de utilizar *Alljoyn Bus* es

que permite que más de dos aplicaciones se puedan comunicar sin la necesidad de lidiar con detalles subyacentes a un sistema distribuido como: manejo de errores en la red, control de transmisión de datos a los distintos dispositivos, manejo de actualizaciones, etc.

- ***Bus Attachment*:** permite que la aplicación en cada dispositivo se pueda comunicar con *Alljoyn Bus.*

- ***Bus interface*:** contiene un grupo de métodos, señales y propiedades asociadas al bus. En la práctica las interfaces son implementadas por el cliente, servidor o *peer*. *Bus interface* provee una manera estándar de declarar una interface que funcione a través de un sistema distribuido.

- ***Bus object*:** Los objetos *Bus Object* existen en los objetos *Bus Attachment* y funcionan como puntos finales de comunicación, estos objetos contienen métodos que pueden ser llamados de forma remota, así mismo pueden emitir señales a través del bus con el fin de transmitir mensajes por el sistema distribuido. En una red Cliente-Servidor el objeto *BusObject* podría ser implementado por el Servidor.

- ***Proxy Object*:** estos objetos son la representación local de los objetos *Bus Object* remotos. Las aplicaciones utilizan a los objetos proxy para realizar llamadas remotas a los objetos de tipo *Bus Object.* En una red Cliente-Servidor el objeto *BusObject* podrá ser implementado por el Cliente.

- ***Event Handler*:** Se encarga de manejar las señales que han sido enviadas. Estas señales son notificaciones asíncronas de eventos o estados que cambiaron en el objeto.

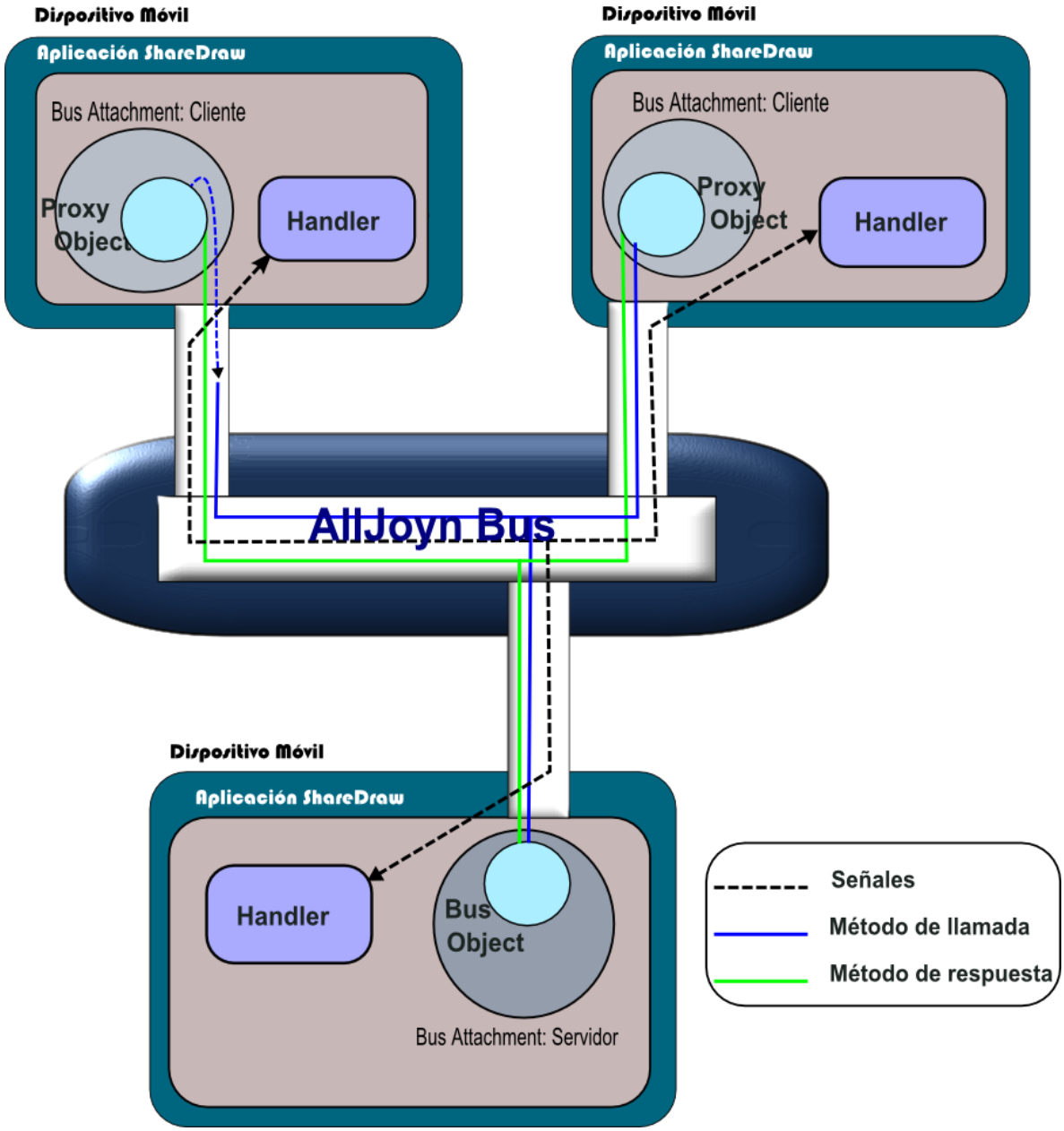


Figura 21, Diseño de la estructura del módulo de colaboración Cliente-Servidor en ShareDraw.

Es importante aclarar que el módulo colaborativo no está integrado a la aplicación *ShareDraw* porque el *framework Alljoyn* es de reciente creación, la documentación no es precisa en cuanto a cómo utilizarla y no cuenta con ejemplos suficientes. Por lo tanto, se experimentó por separado y se creó un módulo independiente llamado *ModuloColaborativoShareDraw*. Este módulo se implementó para dos arquitecturas: Cliente-Servidor y P2P, ofreciendo como funcionalidades: i) que los dispositivos móviles se unan a una sesión que uno de ellos inicie previamente; y ii) la edición: mover, cambiar el tamaño y cambiar el color de una sola figura, la cual es creada por el mismo módulo de colaboración.

Universidad Autónoma Metropolitana Unidad Azcapotzalco

División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Reporte Final

Proyecto Terminal

Aplicación Colaborativa para Dispositivos Móviles con Sistema Operativo Android

Jimena Patricia Alcántara Olivares
Matricula: 207333058

José Daniel López Jaimes
Matricula: 207333480

Asesor: Dra. María Lizbeth Gallardo López
Profesor Investigador
Departamento de Sistemas

Enero 2013

Índice

Introducción

Este proyecto se enfocó a la realización de una aplicación colaborativa llamada *ShareDraw*, la cual permite a un grupo de trabajo intercambiar, editar, modificar o borrar figuras geométricas o texto a través de distintos dispositivos móviles. *ShareDraw* es una aplicación creada para dispositivos móviles que cuenten con Sistema Operativo (S.O.) *Android.*

ShareDraw cuenta con dos espacios de dibujo Espacio Privado y Espacio Público. El Espacio Privado permite que el usuario dibuje y edite objetos sin que nadie más colabore. El Espacio Público permite que el usuario y otros usuarios editen los objetos de manera colaborativa. *ShareDraw* permite crear figuras geométricas así como mensajes de texto breves. Estos objetos pueden ser duplicados o eliminados; las figuras pueden ser modificadas en cuanto a su tamaño y color; y el mensaje de texto puede ser modificado en cuanto a su contenido. El modo colaborativo que incluye la aplicación permite al usuario compartir y editar objetos con otros usuarios estando conectados a una misma red.

El objetivo de este documento es presentar a grandes rasgos el desarrollo de este proyecto terminal. La sección *1* contiene el objetivo general y los objetivos particulares del proyecto. La sección *2* describe los antecedentes del proyecto, haciendo una breve descripción de aquellos proyectos terminales y aplicaciones comerciales que han influenciado en ciertas funcionalidades del actual proyecto. En la sección 3 se da una justificación del proyecto. La sección *4* describe la Metodología de base empleada para el desarrollo del proyecto. En la sección 5 se detalla el desarrollo del proyecto; el cual se divide en dos fases, fase de Diseño y fase de Implementación. Finalmente la sección 6 contiene las conclusiones y perspectivas del proyecto.

1 Objetivos: General y Particulares

1.1 Objetivo General

Diseñar e implementar los espacios de trabajo privado y público de la aplicación "pizarrón compartido" para dispositivos móviles con Sistema Operativo *Android*[1]. Dichos dispositivos estarán interconectados en una red *peer-to-peer*[2] (P2P) para compartir los objetos en el espacio público.

[1] Android es una plataforma completa de código abierto diseñado para dispositivos móviles. Es promovido por Google y es propiedad de la *Open Handset Alliance [2]*.

[2] *Peer-to-peer* o igual-a-igual es un sistema distribuido de comunicación donde cada nodo puede compartir sus recursos solicitando y ofreciendo servicios dinámicamente[3] y no hay división fija de clientes y servidores[4].

1.2 Objetivos particulares:

- Diseñar e implementar un módulo de presentación que incluya un espacio privado y uno público.
- Diseñar e implementar un módulo para gestionar los objetos de los espacios privado y público.
- Diseñar e implementar un módulo para la transición entre los espacios privado y público.
- Diseñar e implementar un módulo de comunicación entre dispositivos móviles bajo una red P2P.

2 Antecedentes

Éste y otros proyectos realizados en la Universidad Autónoma Metropolitana (UAM), Unidad Azcapotzalco, se han enfocado a elaborar diferentes módulos de una aplicación llamada "Pizarrón compartido". Dicha aplicación forma parte de una tesis [1] de maestría elaborada por la M.C. Sánchez Morales, Gabriela inscrita en el dominio del Trabajo Cooperativo Asistido por Computadora, el cual permite a un grupo de colaboradores participar cooperativamente en la producción de entidades compartidas e intercambiar mensajes para coordinarse, aun cuando ellos no puedan reunirse físicamente en el mismo lugar o al mismo tiempo.

2.1 Referencias Internas

A continuación se mencionan los proyectos terminales relacionados con el desarrollo de este proyecto:

"Transición de objetos compartidos entre espacios privado y público en un pizarrón compartido" proyecto terminal del alumno Agustín Morales García [5], al igual que el que se propone en este documento, parten de la idea de diseñar e implementar un espacio de trabajo privado y un espacio de trabajo colaborativo para una aplicación de dibujo llamada "pizarrón compartido". Sin embargo, la aplicación de Morales García se desarrolló para PC y para un simulador de *PDAs*[3]; no se logró llevar la aplicación a *smarthphones*, porque *Java MicroEdition*[4] y sus API's[5] presentaron algunas incompatibilidades con el sistema operativo Android. En la figura 1 se muestran los espacios de la aplicación "pizarrón compartido" para PC.

[3] Del inglés *personal digital assistant o* ayudante personal digital. Es un dispositivo que tuvo un gran auge, ya que fue el primero que contó con la capacidad de realizar tareas parecidas a las de una computadora y de un celular, sin embargo los *smartphones* y tabletas han ido reemplazando esta tecnología debido a su capacidad de alojar un S.O. más avanzado.

[4] Java MicroEdition (J2ME) es un subconjunto de la plataforma Java orientada a proveer una colección de APIs de desarrollo de software para dispositivos con recursos restringidos.

[5] inglés *Application Programming Interface*. Interfaz de programación de aplicaciones. Una API especifica un conjunto de funciones disponibles para un programador de aplicaciones, incluyendo los parámetros que serán pasados a cada función y el valor de retorno esperado por el programador[7].

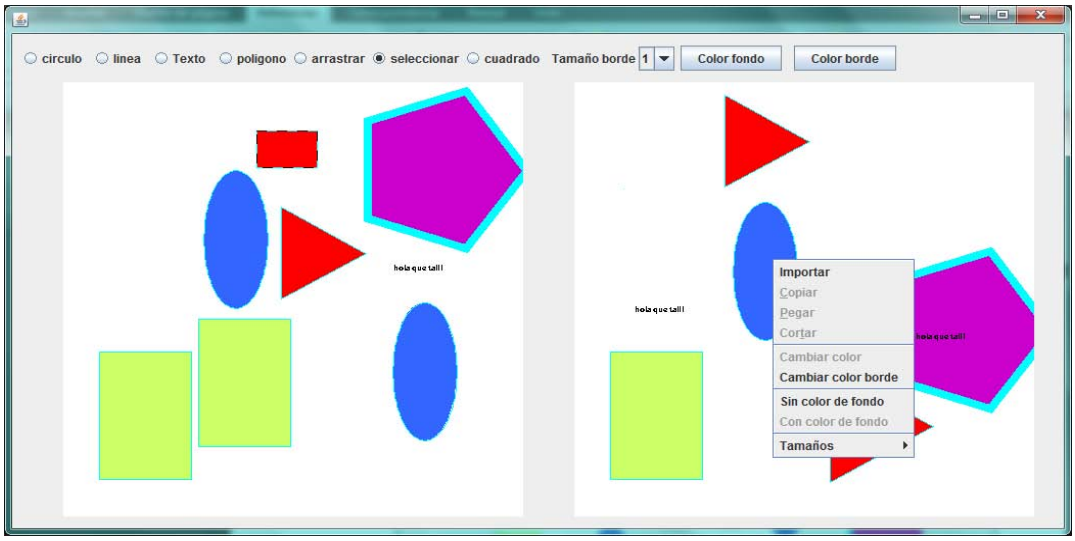


Figura 1. Espacios de trabajo público y privado respectivamente.

"Remodelación plástica de una barra de colaboradores para un espacio de trabajo dentro de un pizarrón compartido" proyecto terminal (en proceso) del alumno Héctor Juárez Cerón. Este proyecto surgió con el objetivo de implementar un algoritmo que rediseñara la barra de herramientas del pizarrón compartido, la cual podría actuar de forma dinámica en función del área de la pantalla del dispositivo.

"Remodelación plástica de la barra de herramientas de un pizarrón compartido" proyecto terminal del alumno Jaime Hernández Cruz [8]. Este proyecto tuvo como fin diseñar e implementar un algoritmo que permitiera desplegar dinámicamente la barra de herramientas de un pizarrón compartido, en función del área del dispositivo de cómputo.

Cada uno de los proyectos antes mencionados son distintos módulos de la aplicación "pizarrón compartido"; y por lo tanto conviene que sean diseñados e implementados de forma independiente. La propiedad de plasticidad en cada uno de ellos depende de las características propias del elemento; por ejemplo no es lo mismo desplegar, en un dispositivo móvil una barra estática como lo es una barra de herramientas, la cual residirá en el mismo dispositivo, a desplegar los objetos (círculo, cuadrado, rectángulo, etc.) creados, inicialmente, en un dispositivo; para posteriormente, ser replicados y adaptados en otros dispositivos.

2.2 Referencias Externas

PREZI es una aplicación comercial que incluye una versión gratuita, la cual está dedicada a la creación de presentaciones en línea con la opción de invitar a otros usuarios a la edición de la presentación de forma colaborativa en tiempo real [9].

Google Docs es un programa para crear documentos en línea con la opción de compartirlos entre otros usuarios y colaborar en tiempo real en la edición [10]. Existe una versión extendida de ésta para el uso en dispositivos móviles para S.O. Android [11].

3 Justificación

En la actualidad muchos proyectos tienen la necesidad de que los miembros de un equipo trabajen a distancia sobre una aplicación, esto implica desarrollar aplicaciones colaborativas donde los miembros puedan compartir y editar el avance general.

Si consideramos que la evolución de los dispositivos móviles permite realizar "tareas robustas"[6], además de permitir a un usuario el acceso en cualquier momento y en cualquier lugar, actualmente es posible migrar aplicaciones para PC a dispositivos móviles, adaptándolas o, en ocasiones, rehaciéndolas para que cumplan los requerimientos del sistema operativo, así como las limitaciones (procesador y memoria) de dichos dispositivos. En proyectos terminales anteriores, tanto de ingeniería como de posgrado, los autores se han basado en estas ideas, aunque no se ha obtenido un avance significativo en cuanto a la portabilidad de estas aplicaciones en dispositivos móviles como *smartphones*[7] y *tablets*[8]. Nuestro proyecto terminal intenta resolver lo asentado anteriormente, rediseñando e implementando una aplicación colaborativa existente capaz de trabajar en una red de dispositivos móviles.

4 Metodología de base empleada en el Proyecto

Para este proyecto empleamos el Proceso Unificado, éste se distingue por utilizar tres conceptos importantes en el desarrollo del software: Dirigido a Casos de uso, Centrado en la Arquitectura, Ser Iterativo e Incremental.

- Es dirigido a Casos de Uso, ya que es parte esencial de un sistema el tomar en cuenta los requerimientos y necesidades del usuario.

- Se centra en la Arquitectura debido a que un sistema debe ser soportado con factores como: la plataforma del software donde se ejecutará, la disponibilidad de ciertos componentes o clases que podrá utilizar, consideraciones de instalación, así como los requerimientos no funcionales.

- Las iteraciones en el desarrollo de un sistema se refieren a pasos en el flujo trabajo, mientras que el crecimiento del sistema se divide en fases de gran alcance, las cuales

[6] Con el término de "tareas robustas" se intenta describir aquellas tareas que gastan recursos considerables como memoria, procesamiento de operaciones, espacio de almacenamiento, interacción con otras aplicaciones como bases de datos y conexiones a redes inalámbricas.
[7] Teléfonos inteligentes.
[8] *Tablet*: dispositivo utilizado como computadora portátil controlado mediante una pantalla multi-táctil.

determinan un incremento. Las fases son: Inicio, Elaboración, Construcción y Transición. Cada fase contiene la realización de las disciplinas: Modelo del Negocio, Requerimientos, Análisis y Diseño, Implementación, Pruebas, Implantación, Manejo de Cambios, Control de Proyecto, Ambiente. En la figura 2 se muestra las iteraciones con las fases previamente descritas.

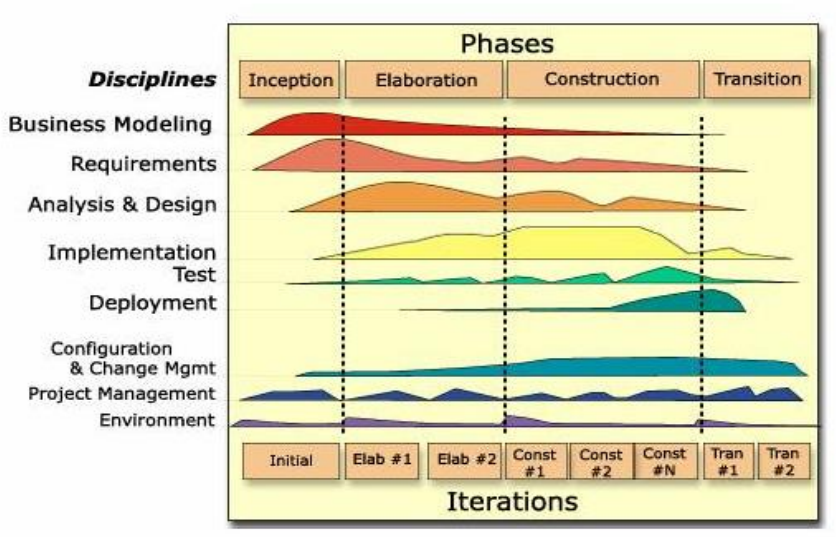


Figura 2. Fases e Iteraciones del Proceso Unificado

5 Desarrollo del Proyecto

En esta sección se describen las etapas de diseño e implementación. La primera sección contiene fragmentos del modelo orientado a objetos que se diseñó para el sistema, a saber: los casos de uso, las clases, el modelo de datos, los diagramas de robustez. La segunda sección describe ciertos segmentos de código directamente relacionados con los elementos de diseño antes mencionados; así como la tecnología empleada en el desarrollo del proyecto.

5.1 Diseño

El diseño será explicado paso a paso y se proporcionarán los diagramas *UML*[9] empleados para modelar la aplicación de dibujo *ShareDraw*.

5.1.1 Diagrama de Casos de Uso

El diagrama general de los casos de uso para nuestra aplicación *ShareDraw* se muestra en la figura 3; los casos de uso son: i) gestionar objetos como son figuras geométricas (círculo,

[9] UML son las siglas de *Unified Modeling Language* (Lenguaje Unificado de Construcción de Modelos), notación (esquemática en su mayor parte) con que se construyen sistemas por medio de conceptos orientados a objetos [12].

cuadrado, triángulo, rectángulo, línea, polígono) y texto; ii) realizar la transición entre espacios público y privado; y iii) colaborar con otro usuario.

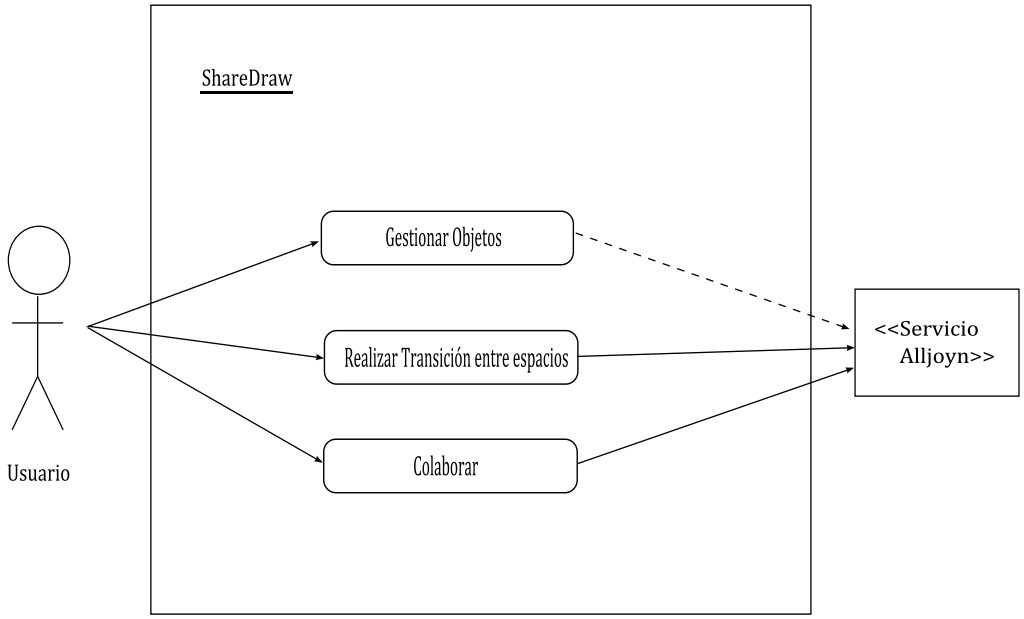


Figura 3.Diagrama de Casos de Uso General

Gestionar Objetos

Este caso de uso tiene como objetivo ofrecer las diversas opciones que podrá realizar el usuario, tanto en el espacio privado como en el espacio público, sobre el objeto u objetos que desea manipular. Las opciones que debe ofrecer son:

- *Crear* objetos (figura o texto)
- *Modificar* características de figura geométrica, tal como el tamaño o el color.
- Modificar características del texto como tamaño y edición de la cadena.
- *Borrar* objetos
- Duplicar objetos

Realizar Transición entre Espacios

Este caso de uso hace referencia a las funciones de compartir e importar un objeto. 1) compartir un objeto desde el espacio privado del usuario al espacio público, permitirá que el resto de los colaboradores manipulen el objeto recién compartido. 2) importar un objeto desde el espacio público al privado permitirá que este usuario manipule de forma exclusiva el objeto recién importado. Además, se debe garantizar que los objetos existentes en cada espacio persistan a pesar de la transición entre los espacios.

Colaborar
Este caso de uso tiene como objetivo permitir al usuario colaborar (vía su dispositivo móvil y una conexión a una red local) en la manipulación de los objetos; de igual forma, debe permitir observar los cambios generados por otros usuarios sobre los objetos en tiempo real.

Se realizaron los escenarios de uso correspondientes a cada uno de los casos de uso antes explicados. Se anexa un ejemplo de escenario de uso en el apéndice A. Luego de realizados los escenarios de uso, empleamos la técnica de tarjetas CRC para construir un modelo conceptual; realizamos también un análisis de robustez; así como un análisis de la arquitectura y del manejo de componentes de Android. Con estos elementos definimos los diagramas de clases. Para mayores detalles sobre el análisis y diseño referirse al documento de diseño del sistema *ShareDraw*.

5.1.2 Diagrama de Clases

El diagrama de clases (ver figura 4) presenta las clases del modelo lógico de la aplicación en color azul; así mismo muestra las clases de *Android* en color verde, las cuales se heredan o se implementan para el funcionamiento de *ShareDraw*.

En la figura 5 se muestra el diagrama de clases del módulo colaborativo de *ShareDraw*. Estas clases heredan o se implementan de clases de la *framework Alljoyn* (en color rojo). Tal y como abordaremos más adelante, el objetivo de utilizar las bibliotecas de *Alljoyn* es para establecer una comunicación adecuada entre dispositivos próximos en una red *p2p* sin la necesidad de recurrir a un servidor intermediario. *Alljoyn* ofrece una solución a problemas como: detección transparente de dispositivos y servicios, funcionalidades de red, el direccionamiento de mensajes. En la sección 5.1.3.2 Clases de Android se presentarán las clases de *Android;* en la sección *5.1.4 Arquitectura de software* se presentarán las clases de *Alljoyn*, y en la sección *5.1.4 Arquitectura de Software* se describirá la lógica de comunicación que se utilizará para el módulo colaborativo y aquellas interfaces o clases de *Alljoyn* necesarias.

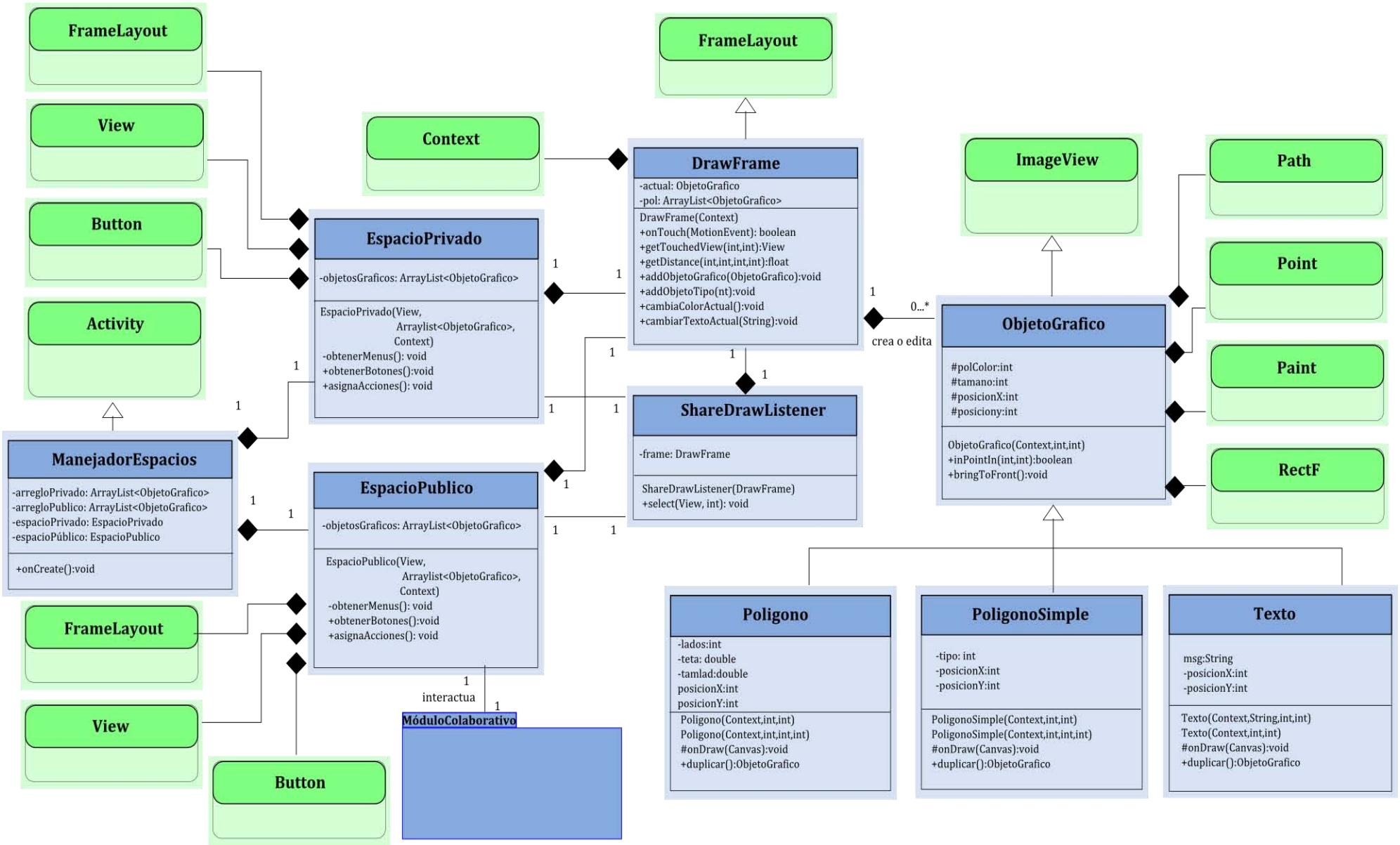


Figura 4. Diagrama de Clases General.

5.1.3 Diagrama de Clases: *ModuloColaborativoShareDraw*

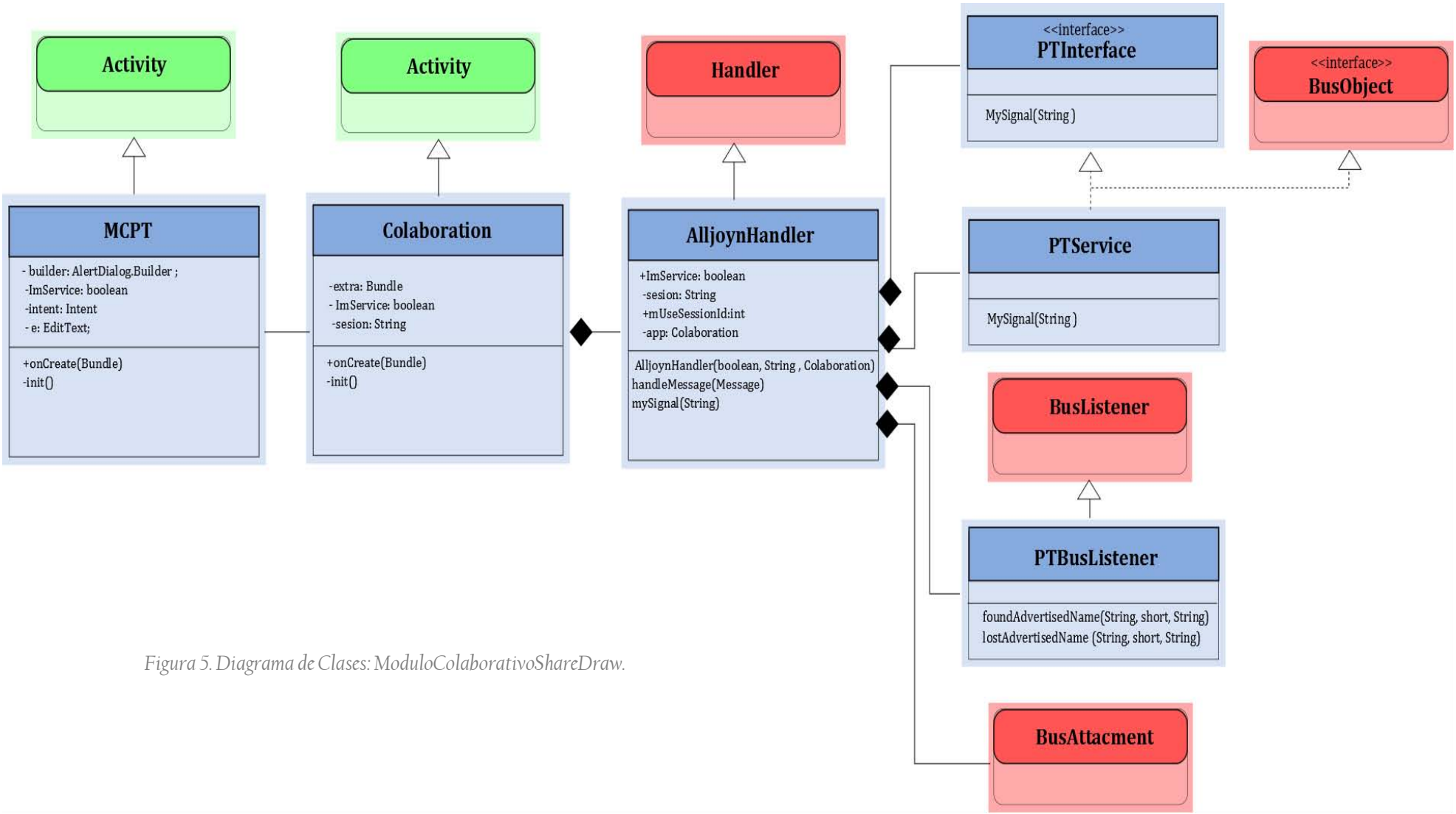


Figura 5. Diagrama de Clases: ModuloColaborativoShareDraw.

5.1.3.1 Clases Android

`Activity`

Una clase `Activity` crea una interfaz de usuario interactiva para una aplicación que va a funcionar en *Android*.

`View`

Esta clase representa el bloque básico de construcción para los componentes de una interfaz de usuario.

`FrameLayout`

Un FrameLayout es una clase que actúa como un marco en la pantalla, el cual puede contener una o más vistas; sin embargo sólo la vista superior es visible completamente, el resto de las vistas se encuentran apiladas por debajo.

`ImageView`

La clase `ImageView` hereda los métodos y atributos de la clase `View.` `ImageView` permite desplegar una imagen a partir de distintos recursos, a saber: imágenes, *bitmaps*, *drawables*[10].

`Button`

Esta clase representa a un botón, sobre el cual se programa una acción específica, a petición del usuario.

`Context`

`Context` es una clase abstracta del sistema *Android* que permite el acceso a recursos específicos de la aplicación; permite además que la aplicación realice solicitudes al sistema Android tales como: lanzadores de actividades, *broadcasting*[11], etc.

`Paint`

La Clase `Paint` contiene la información en cuanto a estilo y color de: i) texto, ii) bitmaps y iii) geometrías a dibujar.

`Path`

La clase `Path` permite trazar caminos geométricos a partir de un vector de puntos, los cuales pueden definir segmentos de líneas rectas, curvas cuadráticas o curvas cúbicas.

`Point`

La clase `Point` representa un punto compuesto de dos coordenadas enteras (x, y).

[10] *Drawable* es una abstracción de *Android* para cualquier cosa que pueda ser dibujado. En este caso se refiere a recursos de la aplicación como son las imágenes de cada uno de los botones.
[11] *Broadcast*: transmisión de los paquetes que serán recibidos por todos los dispositivos en una red.

5.1.3.2 Clases de la Aplicación ShareDraw

`ManejadorEspacios`

La clase `ManejadorEspacios` hereda los atributos y métodos de la clase `Activity`. Por lo tanto la clase `ManejadorEspacios` permite construir la interfaz tanto del espacio público como del espacio privado de la aplicación ShareDraw.

`EspacioPrivado`

Esta clase está compuesta por una clase `View`, la cual permite mostrar la vista del `MenuPrincipal`, la del `EspacioPrivado` y la del `MenuObjeto`.

`EspacioPublico`

Esta clase está compuesta por una clase `View`, la cual permite mostrar la vista del `MenuPrincipal`, la del `EspacioPublico` y la del `MenuObjeto`.

`ShareDrawListener`

La clase `ShareDrawListener` tiene como función principal seleccionar las acciones a realizar en la aplicación. Es importante señalar que estas acciones dependen del espacio en el que se realice la acción (ver Tabla 1). Las acciones son:

Espacio Privado	Acciones	Espacio Público
✓	Crear figura	✓
✓	Crear Texto	✓
✓	Conectarse a una red para colaborar con otros usuarios.	✗
✓	Manual de ayuda breve.	✓
✓	Asignar Color a figura o texto.	✓
✓	Compartir figura o texto del Espacio Privado al Público.	✗
✗	Importar figura o texto del Espacio Público al Privado.	✓
✓	Duplicar figura o texto.	✓
✓	Borrar figura o texto.	✓
✓	Editar texto.	✓

Tabla 1. Acciones posibles a realizar en los espacios respectivos.

La acción conectarse a una red para colaborar con otros usuarios solo puede ser solicitada por un usuario que está trabajando inicialmente en su espacio privado; la acción compartir

figura o texto del Espacio Privado al Público solo puede ser solicitada por el usuario que desea compartir uno o varios objetos, desde su espacio privado; mientras que la acción Importar figura o texto del Espacio Público al Privado solo puede ser solicitada por un usuario que desee modificar una copia de los objetos que se encuentran en el espacio público.

DrawFrame

`DrawFrame` tiene como función principal el crear los objetos gráficos o eliminarlos, así como la edición de éstos en su tamaño, color o contenido.

ObjetoGrafico

`ObjetoGrafico` es una clase genérica que contiene los métodos y los atributos que comparten las clases: `Poligono, PoligonoSimple` y `Texto`.

Poligono

`Poligono` es una clase que hereda de `ObjetoGrafico` y su función es crear polígonos uniformes; es importante destacar que no existen métodos de creación directa de un polígono en los paquetes de gráficos de *Android*; por lo tanto se realizó un algoritmo para dibujar polígonos.

PoligonoSimple

`PoligonoSimple` es una clase que hereda de `ObjetoGrafico` y su función es dibujar figuras simples: línea, círculo, cuadrado y rectángulo.

Texto

Esta clase hereda de `ObjetoGrafico`. Texto es un objeto que contiene una cadena de texto o mensaje, el cual es el que se desea dibujar.

PTInterface

`PTInterface` es una interface necesaria para definir el modelo lógico dentro del sistema distribuido de ShareDraw; describe el comportamiento y los métodos (señales) que serán invocados de manera remota.

AlljoynHandler

`AlljoynHandler` es una clase que hereda de `Handler` (Clase de *Alljoyn*). Es mediante esta clase que se realizan todas las llamadas al bus de comunicación, como son: registrar objetos (*BusObject*), *listener's*[12] (*SessionListener y PortListener*) y *handlers*[13] (*SignalHandler*). Así mismo permite conectarse, desconectarse del bus, crear sesiones y desconectarse de ellas.

[12] *Listener*: la teoría establece que existe un objeto al que llamaremos disparador que realiza alguna acción sobre otro objeto reactivo. El objeto entonces informa a varios *Listener* de que algo ha ocurrido en el disparador. Entonces cada uno de los *listeners* puede realizar alguna acción en base al evento que ocurrió.

Colaboration

Colaboration es una clase que hereda de Activity y su función es detectar los cambios que debe comunicar al bus de comunicación, lo cual es realizado mediante el objeto *AlljoynHandler* creado en esta clase.

MCPT

MCPT es una clase que hereda de *Activity*, su función es presentar un menú de tres opciones al usuario para crear o unirse a una sesión colaborativa o cancelar la opción de colaborar con más usuarios.

PTService

PTService es una clase que implementa a la interfaz *PTInterface*, así como a la interfaz *BusObject* (perteneciente a *Alljoyn*). Esta clase es necesaria para la clase *AlljoynHandler.*

PTBusListener

PTBusListener es una clase que hereda de *BusListener* (perteneciente a *Alljoyn*). Esta clase es responsable de manejar aquellos eventos del Bus de comunicación.

5.1.4 Arquitectura de Software

La arquitectura que se muestra en la figura 6 incluye: sus componentes principales, y la forma en que los componentes interactúan y se coordinan. Se trata de una arquitectura n-capas *(n-layer)* donde las capas se describen a continuación:

Capa de GestionObjetos. En esta capa se encuentran las clases *Drawframe* y *ShareDrawListener* que se encargan de la solicitar la creación y edición de los objetos a dibujar en los espacios de trabajo público y privado. Además, se encuentran las clases encargadas de crear los objetos: *ObjetoGrafico, Poligono, PoligonoSimple* y *Texto*.

Capa de Presentacion. Esta capa se compone de tres paquetes que hacen posible la implementación de una aplicación para Android, a saber: *Drawable, Layout* y *Anim*. El paquete *Drawable* contiene archivos de dibujo necesarios para los íconos de la aplicación. El paquete *Layout* contiene los archivos *XML* útiles para la generación de las interfaces de usuario (ventanas y diálogos) de la aplicación; en el Apéndice B figura B.1 se muestra un archivo *XML* que permite dar animación a un botón, cambiándolo de color una vez que el

[13] *Handler*: manejador de señales.

usuario lo selecciona. El paquete *Anim* contiene los archivos *XML* que definen animaciones, a través de la modificación de las propiedades de un *View*; en el Apéndice B figura B.2 se muestra un archivo *XML* que permite dar animación en el cambio de una vista del *EspacioPrivado* a una vista del *EspacioPúblico* o viceversa. Para el uso de estos archivos es necesario invocarlos y asignarles acciones desde las clases *EspacioPrivado* y *EspacioPublico.* Cabe mencionar que debido a que estas carpetas mencionadas (*Drawable, Layout* y *Anim*) pertenecen a una aplicación de *Android* se encuentran ubicadas en el proyecto, en una carpeta llamada *res.*

Capa de *TransicionEntreEspacios*. Esta capa contiene las clases: *ManejadorEspacios*, *EspacioPrivado* y *EspacioPublico.* La clase *ManejadorEspacios* es la que permite el cambio entre la vista del *EspacioPrivado* y el *EspacioPublico* y viceversa; además permite el paso de Objetos Gráficos de un espacio a otro.

Capa de *ModuloColaborativoShareDraw.* Las clases que componen esta capa son: *PTInterface, Alljoynhandler, MCPT, Colaboration, PTService* y *PTBusListener.* Esta capa lleva a cabo el manejo del bus de comunicación, así como de las señales de alerta de cambios en la aplicación. De igual forma se encarga de la actualización de los datos y la interacción de los objetos entre los distintos espacios públicos, siendo la clase más importante *AlljoynHandler*.

Para esta capa se hizo uso del *framework Alljoyn*, se diseñó un modelo de comunicación (ver figura 7) que permitiera definir la colaboración entre los componentes del *framework* para el módulo colaborativo. Este modelo se diseñó en base a una red Cliente Servidor para comprender los conceptos básicos de *Alljoyn*. Posteriormente, se implementó este modelo en cada dispositivo móvil obteniendo una red *peer-to-peer*; donde, cada dispositivo es, a su vez, Cliente y Servidor en este módulo de colaboración. Los componentes básicos que se requieren para implementar este modelo, son:

- ***Alljoyn Bus***: La abstracción más básica de un sistema con *AllJoyn* es *Alljoyn Bus*, el cual provee un canal de comunicación de alta velocidad que transporta mensajes a través del sistema distribuido. La ventaja de utilizar *Alljoyn Bus* es que permite que más de dos aplicaciones se puedan comunicar sin la necesidad de lidiar con detalles subyacentes a un sistema distribuido como: manejo de errores en la red, control de transmisión de datos a los distintos dispositivos, manejo de actualizaciones, etc.

- ***Bus Attachment***: permite que la aplicación en cada dispositivo se pueda comunicar con *Alljoyn Bus.*

- ***Bus interface***: contiene un grupo de métodos, señales y propiedades asociadas al bus. En la práctica las interfaces son implementadas por el cliente, servidor o *peer*. *Bus interface* provee una manera estándar de declarar una interface que funcione a través de un sistema distribuido.

- ***Bus object***: Los objetos *Bus Object* existen en los objetos *Bus Attachment* y funcionan como puntos finales de comunicación, estos objetos contienen métodos que pueden ser llamados de forma remota, así mismo pueden emitir señales a través del bus con el fin de transmitir mensajes por el sistema distribuido. En una red Cliente-Servidor el objeto *BusObject* podría ser implementado por el Servidor.

- ***Proxy Object***: estos objetos son la representación local de los objetos *Bus Object* remotos. Las aplicaciones utilizan a los objetos proxy para realizar llamadas remotas a los objetos de tipo *Bus Object*. En una red Cliente-Servidor el objeto *BusObject* podrá ser implementado por el Cliente.

- ***Event Handler***: Se encarga de manejar las señales que han sido enviadas. Estas señales son notificaciones asíncronas de eventos o estados que cambiaron en el objeto.

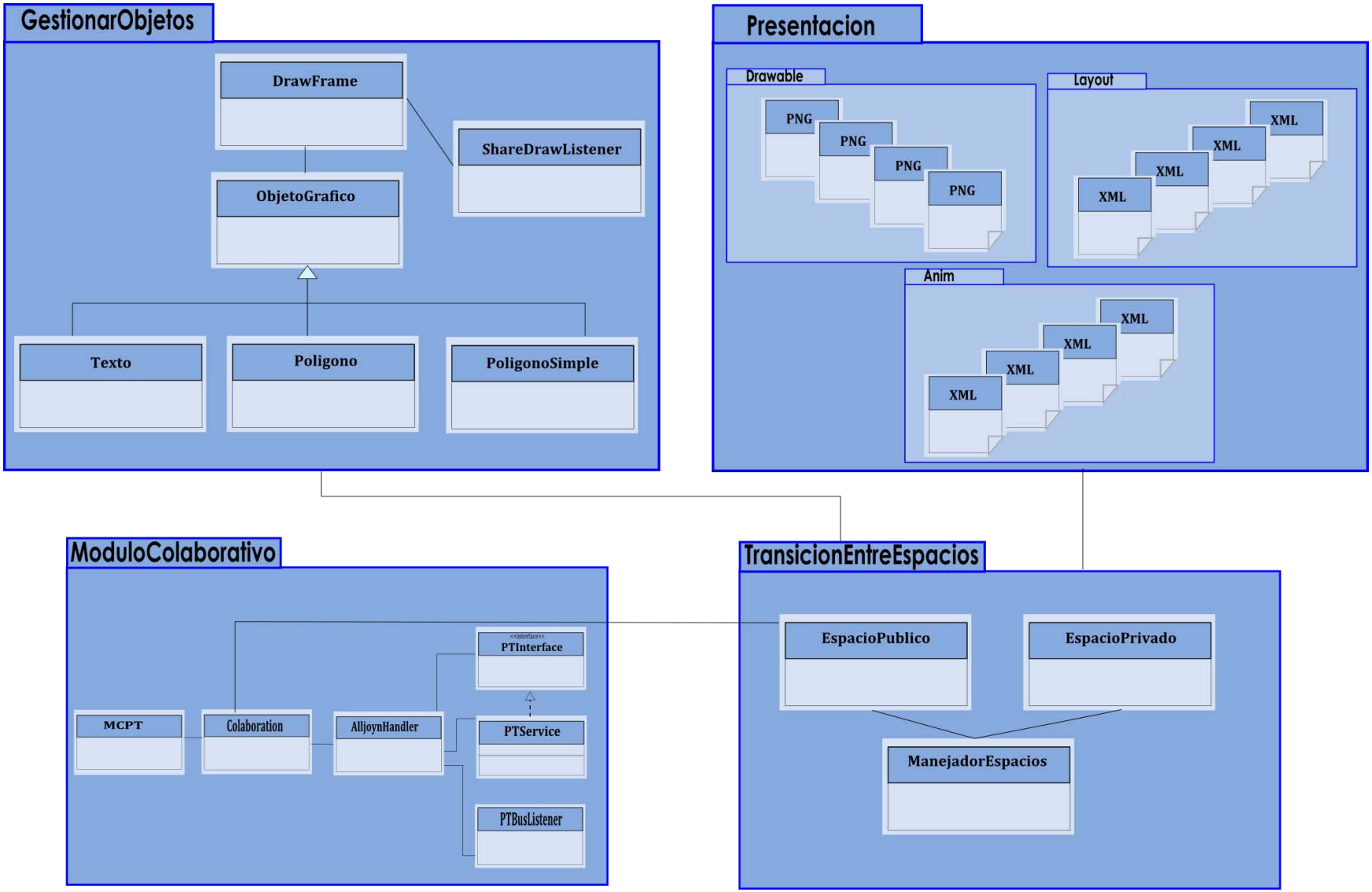


Figura 6. Arquitectura de Software para la aplicación ShareDraw

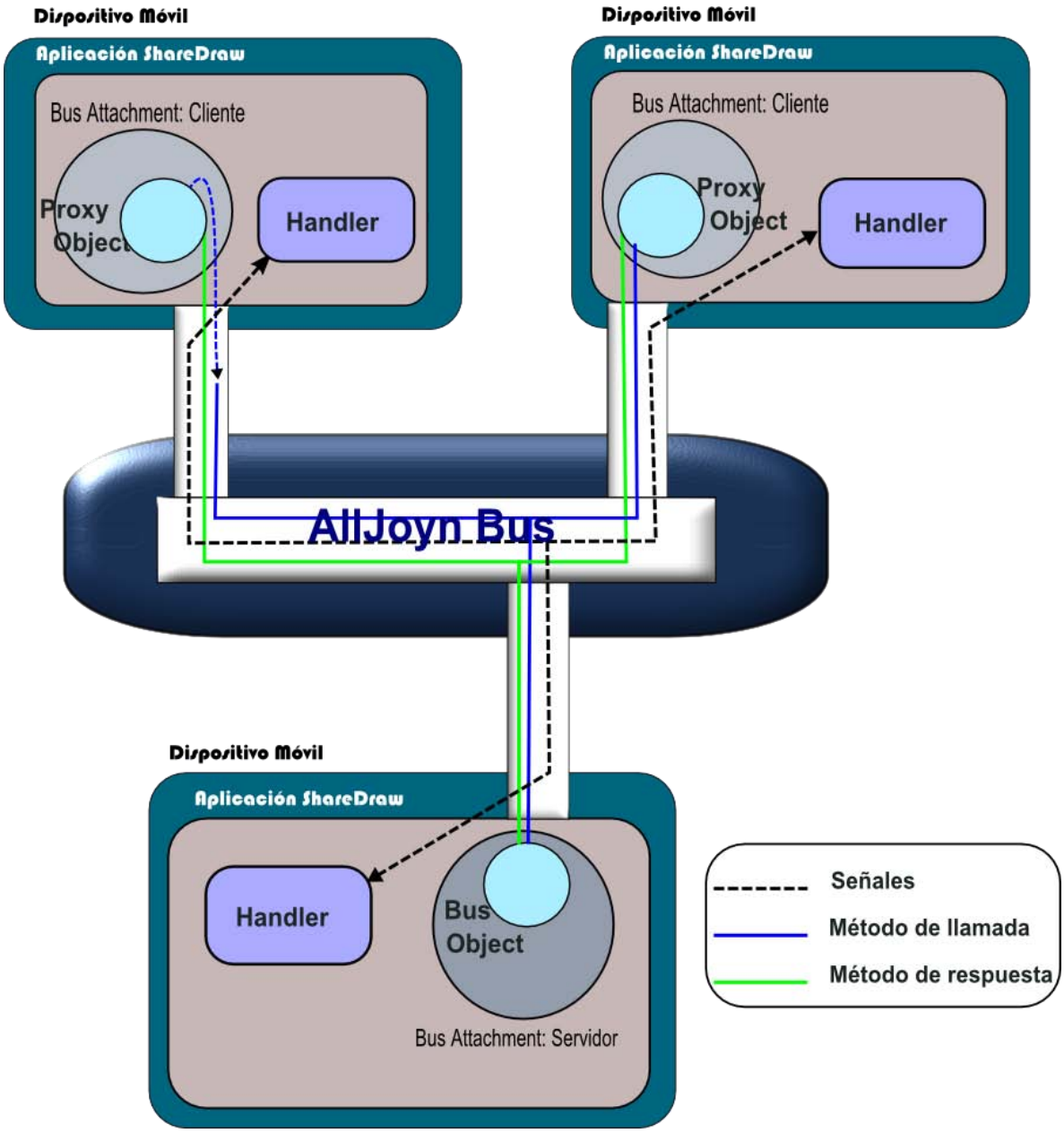


Figura 7. Diseño de la estructura del módulo colaborativo Cliente-Servidor en ShareDraw.

Es importante aclarar que el módulo colaborativo no está integrado a la aplicación *ShareDraw* porque el *framework Alljoyn* es de reciente creación, la documentación no es precisa en cuanto a cómo utilizarla y no cuenta con ejemplos suficientes. Por lo tanto, se experimentó por separado y se creó un módulo independiente llamado *ModuloColaborativoShareDraw*. Este módulo se implementó para dos arquitecturas: Cliente-Servidor y P2P, ofreciendo como funcionalidades: i) que los dispositivos móviles se unan a una sesión que uno de ellos inicie previamente; y ii) la edición: mover, cambiar el tamaño y cambiar el color de una sola figura, la cual es creada por el mismo módulo de colaboración. En las secciones *5.2.3 Ejemplos de implementación*.

5.2 Implementación

Para la implementación de la aplicación *ShareDraw* se utilizó el siguiente *hardware:*

Equipos

- Computadora de escritorio Dell Studio XPS 8100 Procesador Intel core i5 645, 4GB de memoria *RAM, S.O. Windows y Linux.*
- Computadora portátil Dell Inspiron 1545 Procesador Intel Celeron, 3GB de memoria *RAM, S.O. Windows y Linux.*

Dispositivos de Experimentación y Pruebas

- *Smartphone* Samsung I9000 Galaxy
  - Pantalla WVGA de 4 pulgadas
  - Procesador de 1 GHz
  - Sistema operativo *Android* 2.1
  - Wi-Fi
- *Tablet* Samsung Galaxy GT-P1000L
  - 10.1 pulgadas
  - Sistema operativo *Android* 2.3.6.
  - Procesador de doble núcleo a 1 GHz.

- *Smartphone* Xperia MiniPro
  - 3 pulgadas
  - Sistema Operativo *Android* 2.3.4
  - Procesador Snapdragon de 1 GHz

Se implementó la aplicación en los dispositivos móviles antes mencionados por las características y beneficios que ofrecen tanto a nivel de hardware como a nivel de Sistema Operativo. Por un lado, su Hardware está dedicado a realizar multitareas con una velocidad de 1GHz de procesamiento, la cual es superior a la de un móvil telefónico común; además de su capacidad de memoria RAM superior a 500 MB y su capacidad de memoria secundaria superior a 500MB con posibilidad de extenderse, gracias a las tarjetas MicroSD hasta a 32 GB; también cuentan con pantalla táctil y una tarjeta de red. Las características de hardware antes mencionadas hacen que estos dispositivos actúen como una computadora de escritorio. Por otro lado, el Sistema Operativo Android es un sistema de código libre, que engloba un entorno de ejecución basado en Java; esto hace que el desarrollo de las aplicaciones en Android sea un poco más sencillo y accesible para implementar aplicaciones.

El Software utilizado para la implementación, así como para la realización de pruebas fue:

- *IDE*[14] *Java Eclipse (Indigo)*: Éste fue el principal entorno de desarrollo utilizado, porque aporta un plugin (*ADT plugin*) para *Eclipse* que extiende la funcionalidad de éste último y facilita el desarrollo de aplicaciones para *Android*. Entre las funcionalidades de este *plugin* se encuentran:

    - Emulador de *Android*. Permite elegir entre distintas terminales móviles y la versión del sistema operativo.
    - Editores de código para los distintos archivos de configuración (*XML*) que facilitan su comprensión y desarrollo.
    - Interfaces gráficas que permiten el desarrollo de componentes visuales. Ver Apéndice B, figura B.3.
    - El acceso a herramientas de desarrollo de *Android* que permiten: tomar capturas de pantalla, depurar con puntos de parada o ver el estado de los hilos y los procesos corriendo en el sistema.

- *Android SDK*[15] es el kit desarrollo necesario para crear aplicaciones para *Android* y se utilizó en versión de *Windows* y de *Linux*.
- *Framework Alljoyn*, es un marco de desarrollo de código abierto que permite la conexión entre dispositivos en tiempo real, sin la necesidad de intermediarios; utilizando un modelo orientado a objetos, un mecanismo que invoca a un método remoto (*RMI*) e implementando un protocolo de conexión inalámbrica dirigido a una conexión punto a punto.

5.2.1 Características importantes para el funcionamiento de la aplicación ShareDraw

Una vez instalados todos los paquetes de desarrollo es importante conocer la jerarquía del directorio de una aplicación de *Android*, la cual es la misma para *ShareDraw y ModuloColaborativoShareDraw* (ver Figura 8, inciso A), las carpetas que contienen son las siguientes:

src/: La carpeta de contenido (*source folder*) incluye todas aquellas clases de *java* generadas para el funcionamiento de la aplicación *ShareDraw y ModuloColaborativoShareDraw*.

gen/: contiene archivos de Java generados por ADT *plugin*. El *ADT* crea un archivo `R.java`, el cual contiene las referencias a cada uno de los recursos de la carpeta *res*, es mediante este archivo que se pueden invocar los recursos en la clases requeridas.

Android version/ incluye el archivo `Android.jar` que se construyó a partir de la versión 2.2 de *Android*, la cual fue seleccionada cuando se creó el proyecto para desarrollar *ShareDraw y ModuloColaborativoShareDraw*.

[14] Del inglés *integrated development environment* es un entorno de programación, el cual es una aplicación que contiene un editor de código, un depurador, un compilador y un constructor de interfaces gráficas, que proporciona un marco de trabajo amigable para los lenguajes de programación tales como Java.
[15] Del inglés *Standard Edition Kit*

res/: La carpeta recursos (*resource*) contiene los diversos recursos que *ShareDraw y ModuloColaborativoShareDraw* consumen, subdividiéndolos en carpetas dependiendo del tipo de archivo. Los distintos tipos de archivos que consume nuestra aplicación son *anim* dirigido a animaciones para las vistas, *drawable* contiene aquellas imágenes utilizadas en la aplicación, *drawable-hdpi* contiene imágenes para los dispositivos con pantalla de alta definición, *drawable-ldpi* contiene imágenes para los dispositivos con pantalla de baja definición, *drawable-mdpi* contiene imágenes para los dispositivos con pantalla de definición media, *layout* contiene archivos *XML* que definen las interfaz gráfica, *values* contiene archivos *XML* que contienen *Strings*.

Figura 8. a) Jerarquía de las carpetas del Proyecto, b) carpeta libs.

Android.Manifest.xml. El archivo de configuración se llama Android.Manifest.xml; éste es un archivo XML requerido para cualquier aplicación de Android, está localizado en la raíz del directorio del proyecto, y su función es describir los valores globales del proyecto, incluyendo los componentes de la aplicación (actividades, servicios, etc.). Es importante mencionar que al declarar un componente, se declaran al mismo tiempo algunos de sus atributos como son: el momento de iniciar el componente, la posición o el tema que adoptará una vez iniciado el componente.

Finalmente *ModuloColaborativoShareDraw* contiene una carpeta más, necesaria para implementar la colaboración en conjunto con el *framework Alljoyn.*

libs/: esta carpeta se compone de dos paquetes de librerías de *Alljoyn: Alljoyn.jar y liballjoyn_java.so (ver figura 8 inciso b).*

5.2.2 Descripción Técnica: Clases de la Aplicación *ShareDraw*

ManejadorEspacios

`ManejadorEspacios` hereda de la clase *Activity* algunos métodos que se manejan en su ciclo de vida de, estos métodos son: *`onCreate`*, *`onPause`*, *`onResume`*, *`onDestroy`*. Al crear una clase que hereda de *`Activity`* es imprescindible implementar el método *`onCreate`*; el cual se ejecuta una vez creada una instancia de `ManejadorEspacios`. En el método *`onCreate`* es donde se cargan y se manejan los cambios entre vistas *`(View)`* de un *`EspacioPrivado`* y de un *`EspacioPublico`*. Así mismo tiene la función de importar o compartir los objetos gráficos de un espacio a otro, por medio de la actualización de los arreglos de objetos gráficos y del marco de dibujo (*`DrawFrame`*) pertenecientes a cada espacio.

EspacioPrivado y EspacioPublico

Los menús `MenuPrincipal`, `MenuObjeto` contenidos en `EspacioPrivado` *y* *`EspacioPublico`* serán visibles a partir de los disparadores correspondientes durante el uso de *ShareDraw*. Un ejemplo de disparador sería cuando el usuario oprime la tecla menú del dispositivo. Una vez oprimida se mostrará el *`MenuPrincipal`*. Cada menú se compone de distintos botones, los cuales se obtienen a partir los *`drawables`* correspondientes.

Otra función del *`EspacioPrivado`* y *`EspacioPublico`* es el de modificar una bandera en el objeto seleccionador *`ShareDrawListener`*. Esta bandera cambia de un estado de espera a la acción seleccionada a través de un botón. Existen diversas acciones a realizar, las cuales se comentarán más adelante. Un ejemplo de una acción perteneciente al espacio privado es dibujar una figura. El botón de dibujar figura se encuentra en el menú principal; por tanto, una vez seleccionado, éste asignará *`SELECCION_FIGURA`* en la bandera del objeto *`ShareDrawListener`*.

ShareDrawListener

`ShareDrawListener` actúa como un *switch* de selección. La opción o bandera que se reciba por medio del método *`select`* disparará la acción a realizar en el *`DrawFrame`*. Las opciones son las siguientes:
`SELECCION_FIGURA`, *`SELECCION_TEXTO`*, *`SELECCION_CONEXION`*, *`SELECCION_AYUDA`*, *`SELECCION_COLOR`*, *`SELECCION_EDITATEXTO`*, *`SELECCION_COMPARTIR`*, *`SELECCION_DUPLICAR`*, *`SELECCION_BORRAR`* *y* *`SELECCION_IMPORTAR`*.

5.2.3 Descripción Técnica: Clases del *ModuloColaborativoShareDraw*

MCPT

MCPT es una actividad que contiene tres opciones: crear sesión colaborativa, unirse a una sesión existente, cancelar el unirse a la colaboración. Se validarán las entradas para los dos primeros casos, el nombre de una sesión no podrá exceder más de 20 caracteres, así mismo si se deseara iniciar una sesión y no se encontrara activo el servicio de Alljoyn (necesario para colaborar en el espacio público), la aplicación retroalimentará al usuario con un mensaje de error.

Colaboration

Colaboration es una clase que hereda de *Activity*. Es en esta clase donde se carga de manera estática el paquete de clases del *framework* de *Alljoyn*, así una vez que reconoce el paquete de *Alljoyn*, se podrá crear un objeto de tipo *AlljoynHandler* llamado *AJHandler*. Mediante los métodos de *AJHandler* será posible transmitir al bus de comunicación los cambios que se deben realizar en los demás dispositivos conectados en la red.

PTInterface

Los métodos que contiene esta interfaz son aquellos que permitirán actualizar los cambios en los objetos: *mover, cambiarTamaño, cambiarColor.*

AlljoynHandler

AlljoynHandler es una clase que hereda de Handler. Al implementar el método *handleMessage* es capaz de manejar los mensajes que sean transmitidos, los mensajes que dispararán una acción son: *CONNECT, DISCONNECT, JOIN_SESSION, CREATE_SESSION, SEND_MESSAGE.* *SEND_MESSAGE* permitirá transmitir los datos que hayan cambiado a los demás dispositivos. En esta clase se crearon dos clases anidadas requeridas para la transmisión y actualización de los datos *PTService* y *PTBusListener*. La primera es necesaria debido a que implementa los métodos de la interfaz *PTInterface*; la segunda clase contiene aquellos eventos que requiere manejar del bus de comunicación, en este caso, *foundAdvertisedName y lostAdvertisedName.*

Un aspecto importante a destacar es la integración del módulo colaborativo con la aplicación ShareDraw. Para lograrlo, es necesario crear un método en la clase *AlljoynHandler* para cargar todos los objetos gráficos de la sesión en la lista de objetos cada vez que un usuario o se una a la sesión. También es necesario crear una clase que se encargue de dividir el mensaje de la señal enviada entre los dispositivos y que decida qué objeto está siendo editado.

5.2.4 Ejemplos de Implementación

Antes de mostrar la implementación de algunos módulos, describiremos el archivo de configuración *AndroidManifest.xml*. Este archivo presenta la información esencial que el

sistema Android debe conocer sobre la aplicación ShareDraw antes de ejecutarla. La configuración incluye:

- Declarar el nombre del paquete de Java para la aplicación.
- Declarar la versión mínima de la API de Android que requiere ShareDraw, a saber la versión 8.
- Describir los componentes de ShareDraw.

Estructura de *AndroidManifest.xml*

El Código 1 muestra la estructura general de *AndroidManifest.xml* a través de etiquetas, donde cada etiqueta refiere a un elemento o conjunto de elementos que componen a ShareDraw, a saber:

- *<application>* este elemento contiene los componentes que va a emplear una aplicación. En este caso ShareDraw, así como sus principales atributos, por ejemplo el ícono de la aplicación y su nombre.
  - *<activity>* este elemento declara una actividad (Activity) en una aplicación. En este caso se declaran tres actividades para *ShareDraw*. A las tres actividades se les atribuyen características como es el tema de la aplicación y la orientación de la pantalla; sin embargo, la actividad llamada *ShareDrawActivity* contiene un *IntentFilter.*
    - *<IntentFilter>* describe la acción que solicita el usuario y cuándo debe de ser iniciada. En *ShareDraw* la acción es la actividad inicial que se activará cuando el usuario seleccione el ícono de inicio que se encuentra en el "menú de aplicaciones" del dispositivo con *Android*. En el apéndice B figura B.4 se muestra la pantalla de inicio de la aplicación *ShareDraw*, la cual es la vista gráfica de la actividad *ShareDrawActivity*.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.jimdan.pt" android:versionCode="1" android:versionName="1.0">

    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/ic_launcher"
        android:label="@string/app_name">
        <activity android:name=".ShareDrawActivity" android:label="@string/app_name"
            android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen"
            android:screenOrientation="landscape">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name=".EspacioPublico"
            android:theme="@android:style/Theme.Light.NoTitleBar.Fullscreen"
            android:screenOrientation="landscape"></activity>
        <activity android:name="ManejadorEspacios"
            android:screenOrientation="landscape"
            android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"></activity>
    </application>

</manifest>
```

Código 1. AndroidManifest ShareDraw.

5.2.4.1 Iniciando Aplicación

Una vez que el usuario da inicio a la aplicación *ShareDraw*, sea para dibujar en un modo individual (privado) o en un modo colaborativo (público), se le mostrará el espacio de dibujo correspondiente. Para este ejemplo se considera que el usuario seleccionó el modo individual; la figura 9 muestra la pantalla del `EspacioPrivado`. En el apéndice B figura B.5 muestra la pantalla para el modo colaborativo, `EspacioPublico`.

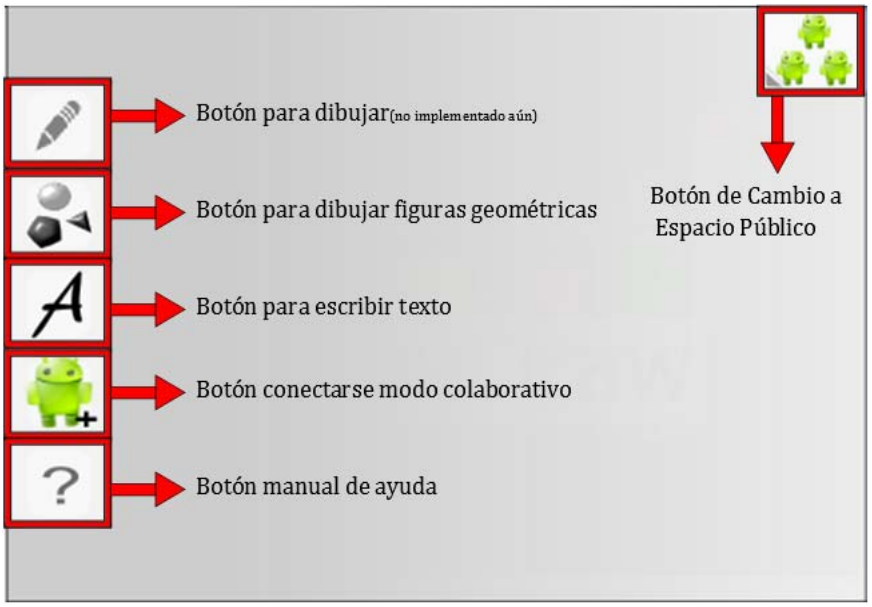


Figura 9. Primera Pantalla en EspacioPrivado.

Clase: *ManejadorEspacios*

El método *onCreate* (ver código 1) establece la vista inicial del *EspacioPrivado* (como se puede observar en la figura 8) o del *EspacioPublico* dependiendo de la selección del usuario; esto se logra mediante la animación[16] *flipper*. La última sentencia del código muestra la inicialización de la vista gráfica de EspacioPrivado a partir del *layout* identificado como *R.id.privado*.

```
public class ManejadorEspacios extends Activity {
protected void onCreate(Bundle savedInstanceState) {
    setContentView(R.layout.flipper);
                    :
    this.espacioPrivado = new EspacioPrivado(findViewById(R.id.privado),
            arregloPrivado, this);
```

Código2. ManejadorEspacios, método onCreate.

5.2.4.2 Mostrando Menú Principal de Espacio Privado

Clase: *ManejadorEspacios*

El método *onKeyUp* (ver código 2), heredado de *Activity* en la clase *ManejadorEspacios*, disparará una acción cuando el usuario presione el botón de menú del dispositivo. La acción será mostrar el menú principal (ver figura 10) y el botón de cambio entre espacios, dependiendo de la vista en que se encuentre (*EspacioPrivado* o *EspacioPublico*).

```
public boolean onKeyUp(int keyCode, KeyEvent event) {

    if (keyCode == KeyEvent.KEYCODE_MENU) {
        if (espacioPrivado.isShown())
            espacioPrivado.showMenusGenerales();
```

Código 3. Método onKeyUp, clase ManejadorEspacios

[16]Una animación se refiere a implementar efectos de transparencia o de rotación o de translación sobre objetos gráficos de Android como lo es *View*.

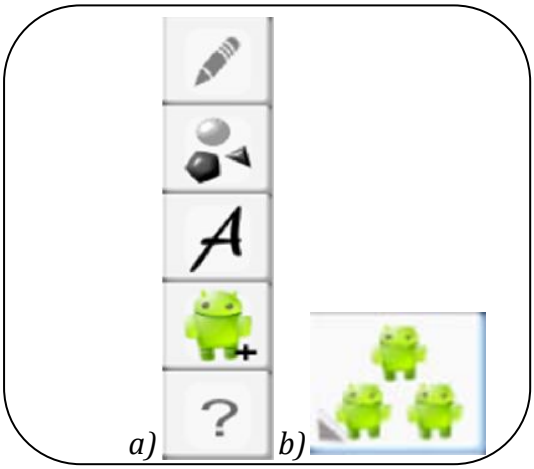


Figura 108. En el inciso a) se muestra Menú Principal del Espacio Privado, en el inciso b) se muestra el botón de cambio entre espacios.

En el apéndice B, figura B.5 se puede observar el menú principal del EspacioPublico.

5.2.4.3 Seleccionar una opción en Espacio Privado: Dibujar Figura

Clase: EspacioPrivado

En el método `asignaAcciones` de la clase `EspacioPrivado,` el objeto seleccionador (`ShareDrawListener)` se crea (ver Código 3). `ShareDrawListener` contiene un método `select` cuya función es modificar la bandera `seleccion` que especifica la acción a realizar en la próxima interacción del usuario con la aplicación.

```
public void asignaAcciones() {
    final ShareDrawListener l = new ShareDrawListener(drawFrame);

    botonFigura.setOnClickListener(new View.OnClickListener() {
        public void onClick(View v) {

            l.Select(v, ShareDrawListener.SELECCION_FIGURA);
            despareceMenus();
        }
    });
```

Código 4. Método asignaAcciones, clase EspacioPrivado.

A partir de que el usuario presione un botón del menú principal, se seleccionará en el `ShareDrawListener,` la acción a realizar. Las acciones a realizar son las antes descritas en el apartado de Diagrama de Clases: crear figura, crear texto, editar texto, borrar objetos, etc. Si la bandera `seleccion` es igual a `SELECCION_FIGURA`, se realizarán los siguientes pasos: 1) crear una variable tipo `AlertDialog`. 2) Se inicializa una vista con el `layout` que se desea

mostrar en el diálogo (`R.layout.eleccion_figuras`). 3) Se crea un objeto `builder`, éste será el que asigne los atributos del objeto `aDialog`, entre ellos el título, el ícono y la vista que tendrá. 4. Una vez inicializado el `aDialog`, se muestra en la pantalla.

```
public void Select(View v, int seleccion) {
    switch (seleccion) {
            case SELECCION_FIGURA: {
        // Alert Dialog Elección de Figuras
        final AlertDialog aDialog;
        LayoutInflater factory = (LayoutInflater) v.getContext()
                .getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
        // carga layout de las figuras AlertDialog de las figuras
        View eleccionFig = factory.inflate(R.layout.eleccion_figuras,
                (ViewGroup) v.findViewById(R.id.MenuEleccionFiguras));
        AlertDialog.Builder builder = new AlertDialog.Builder(
                v.getContext());

        builder.setTitle("Elige Figura");
        builder.setIcon(R.drawable.ic_launcher);
        builder.setView(eleccionFig);
        aDialog = builder.create();
        aDialog.show();
```

Código 5. Método Select, clase ShareDrawListener.

En la figura 11 se muestra el diálogo para seleccionar la figura a dibujar.

Figura 11. Diálogo de selección de figura a dibujar.

5.2.4.4 Dibujando una Figura en DrawFrame: Poligono

Clase: `DrawFrame`
Esta clase define el marco de dibujo donde se mostrarán los objetos gráficos creados.
El método que permite agregar un nuevo ObjetoGrafico, es `onTouch`, el cual recupera la posición (X,Y) tocada por el usuario donde desea que se dibuje el ObjetoGrafico que seleccionó previamente. Una vez que se obtiene la posición X y Y, es posible crear el nuevo ObjetoGrafico (las posiciones X y Y forman parte de los atributos de un ObjetoGrafico); para finalmente, agregarlo al DrawFrame del espacio de dibujo actual (ver Código 5).

```
public boolean onTouch(View v, MotionEvent event) {
    // -1=nada, 0=linea, 1=circulo, 2=rectangulo, 3=triangulo, 4=cuadrado,
    // 5, 6, 7, 8,...=poligonoregular
    // 1000 = texto
    if (bandera) {
        bandera = false;
        switch (TIPO) {
        case 0:
            addObjetoGrafico(new PoligonoSimple(this.getContext(), 0,
                    (int) event.getX(), (int) event.getY()));
            break;
        case 1:
            addObjetoGrafico(new PoligonoSimple(this.getContext(), 1,
                    (int) event.getX(), (int) event.getY()));
            break;
```

Código 5. Método onTouch, clase DrawFrame

Tomando el caso en que la figura a dibujar sea un polígono, se hace uso de la clase `Poligono` y de su método `onDraw`, el cual permite dibujarlo.

Clase: `Poligono`
La clase `Poligono` se utiliza para crear polígonos regulares. En su constructor contiene los atributos tales como: el número de lados, `teta` y `tamlad`, los cuales permiten definir un polígono. Cabe aclarar que no existe ningún método en el *API* de *Android* que dibuje un polígono con número de lados superior a 4. Por tanto, se tuvo que implementar un algoritmo para generar cualquier polígono regular, en el Código 6 se muestran las sentencias a realizar para crear los polígonos.

```
protected void onDraw(Canvas canvas) {
    //el vector de puntos tendrá el tamaño ddel numero de lados asignados al poligono.
    vectorDePuntos = new Point[lados];
    //Se crea el "camino" el cual  contendrá los puntos calculados para generar el poligono.
    polPath = new Path();
    // se calculan los puntos matematincamente en base al numero de lados. Guardándolos en en el vetor de puntos
    for (int j = 0; j < lados; j++) {
        vectorDePuntos[j] = new Point((int) (this.posicionX + tamano
                * Math.cos(teta)), (int) (this.posicionY + tamano
                * Math.sin(teta)));
        teta += tamlad;
    }
    //se asignan los puntos por donde el polpath debe crearse
    polPath.moveTo(vectorDePuntos[0].x, vectorDePuntos[0].y);
    for (int j = 1; j <= (lados - 1); j++) {
        polPath.lineTo(vectorDePuntos[j].x, vectorDePuntos[j].y);
    }
    //se dibuja el poligono
    super.onDraw(canvas);
}
```

Código 6. Método onDraw, el cual contiene el algoritmo para crear polígonos regulares.

Su funcionamiento es el siguiente: un polígono se puede dibujar a partir de una circunferencia. Empleamos la variable `tamlad` la cual contiene la distancia que deberá tener cada una de las aristas del polígono. Después, es necesario calcular en qué posiciones estarán los nodos de cada arista, para esto sumamos a la coordenada X el tamaño que debe tener la figura y esto se multiplica por el coseno de teta (que al principio vale cero); de igual forma a la coordenada Y se le suma el tamaño que debe tener la figura y se multiplica por el seno de teta. El multiplicar la suma por el seno de teta o coseno de teta respectivamente permite dibujar los puntos de intersección entre la circunferencia y el nodo de cada una de las aristas (ver figura 12). Cada uno de estos puntos es agregado a un vector de puntos. Una vez obtenidos todos los puntos se dibujarán con el método `lineTo` del objeto `polPath`, el cual va trazando el camino de puntos mediante líneas.

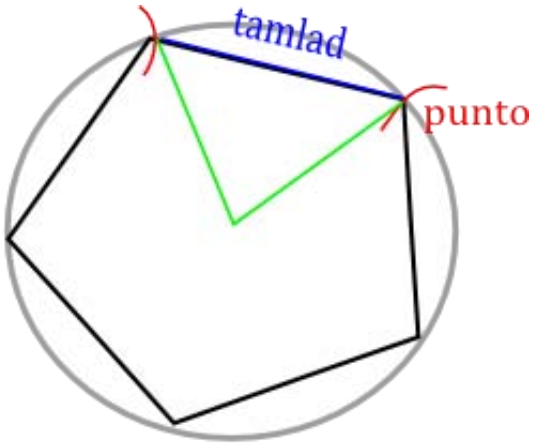


Figura 12. Muestra gráfica de algunos conceptos necesarios para dibujar polígonos regulares. En rojo se muestran los puntos a unir por la arista llamada tamlad.

El resultado final de dibujar una figura en el espacio de dibujo se muestra en la figura 13.

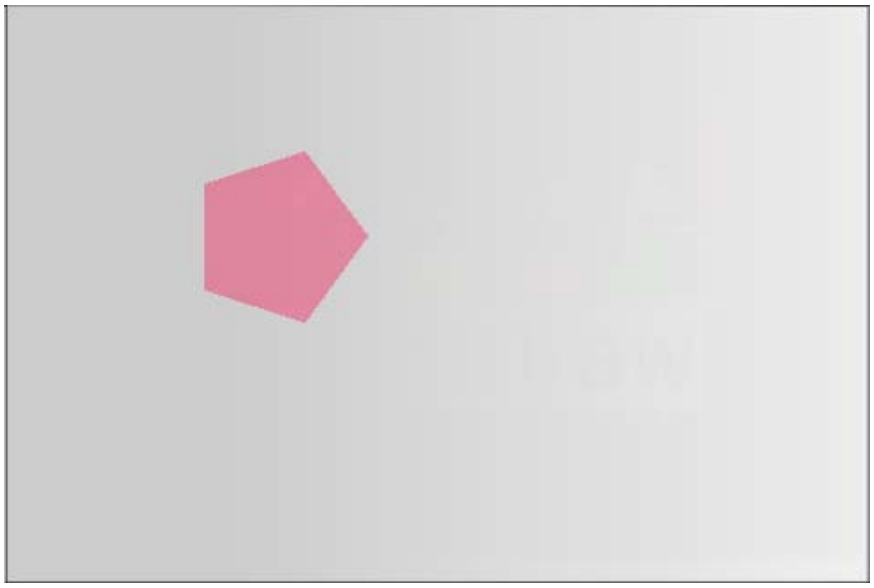


Figura 13. ObjetoGrafico agregado a DrawFrame.

5.2.4.5 Aparecer Menú Objeto

Clase: `EspacioPrivado ó EspacioPublico`
Estas dos clases implementan la interfaz `OnLongClickListener` y su método `onLongClick`.

Una vez detectado un `onLongClick`[17] sobre las *View's* de cada espacio, lo que se hace es verificar una condición (ver Código 7). Esta condición tiene que asegurarse de que la vista que está siendo seleccionada tenga la referencia a una instancia de `ObjetoGrafico` y no una referencia vacía (*null*). Si la referencia es una instancia de `ObjetoGrafico`, entonces se debe asegurar que el `ObjetoGrafico` no se esté desplazando por el `DrawFrame`; es decir, que el objeto se encuentre estático para que aparezca el menú del objeto; cumpliendo estas dos condiciones, se mostrará el `MenuObjeto` del `ObjetoGrafico` seleccionado.

[17] El método *onLongClick* permite detectar un "toque largo" sobre la vista que lo implementa. Una vez que se detecte, se dispararán las acciones contenidas en el método.

```
public boolean onLongClick(View v) {

    if(drawFrame.getActual() instanceof ObjetoGrafico && drawFrame.getActual() !=null){
     actual = drawFrame.getActual();
        if ( !actual.isInmove()) {
            despareceMenus();
            if (!(actual instanceof Texto))
                botonEdittext.setVisibility(View.GONE);
            else
                botonEdittext.setVisibility(View.VISIBLE);
            showMenuObjeto();
        }
    }
    else if(MenuObjeto.isShown())desapareceMenuObjeto();
    return true;
}
```

Código 7. Método onLongClick, clase EspacioPrivado o EspacioPublico.

El `MenuObjeto` difiere según el objeto gráfico, en la figura 14 se muestra el `MenuObjeto` para los objetos: `PoligonoSimple` ó `Poligono`; mientras que en la figura 15 se muestra el `MenuObjeto` para objetos tipo `Texto.`

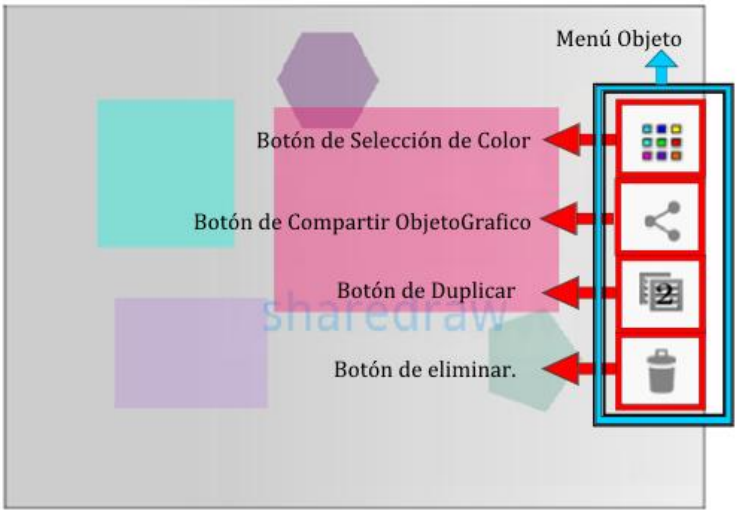


Figura 14. Menú Objeto: Poligono o PoligonoSimple en EspacioPrivado

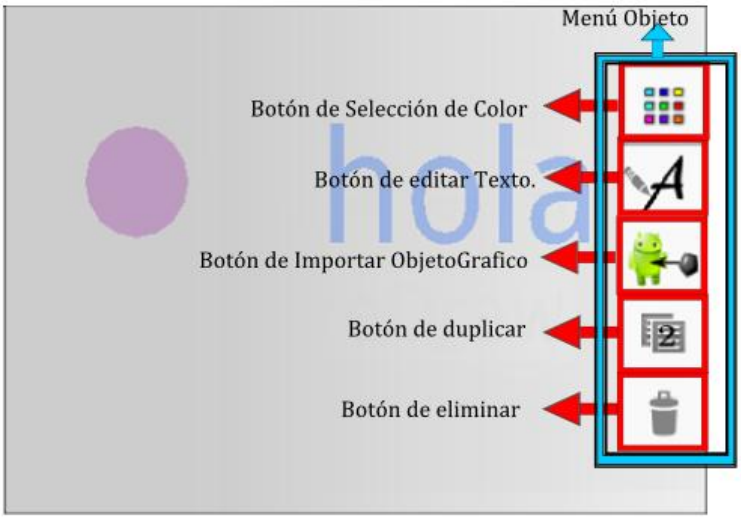


Figura 15. Menú Objeto: Texto en EspacioPublico.

5.2.4.6 Detectando Multitouch

Clase: DrawFrame

La clase DrawFrame implementa el método o`nTouch` para detectar un toque sobre alguna de las vistas contenidas en esta clase; mediante este método se obtiene un objeto `MotionEvent`. `MotionEvent` permite obtener información acerca del evento que se acaba de detectar, a saber: la posición del punto que fue tocado y el tipo de movimiento realizado por el usuario. Existen tres movimientos o acciones que se pueden identificar:

- `ActionDown`, cuando el usuario toca la pantalla por primera vez con uno varios dedos, el `actionDown` los detecta como puntos iniciales.
- `ActionMove`, detecta uno múltiples puntos estáticos o en movimiento sobre la pantalla.
- `ActionUp`, se dispara cuando el usuario deja de tocar la pantalla.

Siguiendo con nuestro ejemplo, suponiendo que el movimiento que realizó el usuario es de tipo `ActionMove`, la acción que se dispara puede ser: 1) el cambio de tamaño de un `ObjetoGrafico` ó 2) el movimiento de un `ObjetoGrafico`, como se muestra en el código 8. La acción depende del número de puntos tocados en la pantalla, los cuales son asignados a la variable `Count`. Sí `Count` es igual a uno, entonces se moverá el objeto gráfico siguiendo las coordenadas obtenidas por el `MotionEvent`. En cambio si la cuenta es igual a dos, entonces cambiará de tamaño el objeto gráfico. Para cambiar el tamaño del objeto, se realizarán los siguientes pasos: 1) Se calcula la distancia entre los dos puntos tocados, 2) sí la distancia `actual` entre los dos puntos es menor a la distancia anterior (`distancia_entre2`), el tamaño que tenía el objeto gráfico se reducirá en tres; pero sí la distancia `actual` es mayor, el tamaño del objeto gráfico aumentará en tres. En el Apéndice B, Figura B.6 y Figura B.7 muestran un ejemplo de un gesto *multitouch*, el cual permite cambiar de tamaño a los objetos gráficos.

```
private void actionMove(int count, MotionEvent event) {
    switch (count) {

    //moviendo ObjetoGrafico
    case 1:
        if (actual != null && actual.isTocado()) {
            actual.setPosicionXY((int) event.getX(), (int) event.getY());
            actual.setInmove(true);
        }
        break;
        //Cammbiando tamaño ObjetoGrafico
    case 2:
        if (actual != null && actual.isTocado()) {
            int distancianueva = (int) getDistance((int) event.getX(0),
                    (int) event.getY(0), (int) event.getX(1),
                    (int) event.getY(1));
            // int diff = distancianueva - distancia_entre2;
            actual.setTamano((distancianueva - distancia_entre2 < 0) ? (actual
                    .getTamano() - 3) : (actual.getTamano() + 3));
            distancia_entre2 = distancianueva;
        }
        break;
    default:
        break;
    }
    this.invalidate();
}
```

Código 8. Método ActionMove, clase DrawFrame.

5.2.4.7 Transición de un Objeto Gráfico entre espacios

Clase: `ManejadorEspacios`
En su método `onCreate` asigna un `onClickListener` al botón `Importar`, cuando éste sea pulsado, se duplicará el objeto gráfico seleccionado agregándolo al `DrawFrame` del `EspacioPrivado` mediante el método `addView`, como se muestra en Código 9.

```
this.espacioPublico.getBotonImport().setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        espacioPrivado.getFrame().addView(espacioPublico.getFrame().getActual().duplicar());
        espacioPublico.showMenuObjeto();
    }
});
```

Código 9. Fragmento de método onCreate de la clase ManejadorEspacios, el cual permite importar un ObjetoGrafico del EspacioPublico al EspacioPrivado.

Para compartir un objeto gráfico del `EspacioPrivado` al `EspacioPublico` se hace de la misma forma, ver Código 10.

```
this.espacioPrivado.getBotonExport().setOnClickListener(new OnClickListener() {

    public void onClick(View v) {
        espacioPublico.getFrame().addView(espacioPrivado.getFrame().getActual().duplicar());
        espacioPrivado.showMenuObjeto();
    }
});
```

Código 10. Fragmento de método onCreate de la clase ManejadorEspacios, el cual permite compartir un ObjetoGrafico del EspacioPrivado al EspacioPublico.

5.2.4.8 Módulo colaborativo: Conectar al bus de Alljoyn.

Proyecto: `ModuloColaborativoShareDraw.`

Clase: `AlljoynHandler.`

Las llamadas al bus de Alljoyn se realizan en `AlljoynHandler` que hereda de `Handler` el método `handleMessage` donde por selección (sentencia switch/case) se determina la acción a realizarse entre el bus y la aplicación. El Código 11 muestra la parte de la conexión al bus.

```
@Override
public void handleMessage(Message msg) {
    switch (msg.what) {
    case CONNECT:
        mBus.registerBusListener(mBusListener);
        status = mBus.registerBusObject(mPTService, OBJECT_PATH);
        if (Status.OK != status) {
            app.UIHandler.sendMessage(app.UIHandler.obtainMessage(
                    Colaboration.TO_SCREEN, "Error en registerBusObject"));
            app.finish();
            return;
        }
        status = mBus.connect();
        if (Status.OK != status) {
            app.UIHandler.sendMessage(app.UIHandler.obtainMessage(
                    Colaboration.TO_SCREEN, "Error en Connect"));
            app.finish();
            return;
        }
        status = mBus.registerSignalHandlers(this);
        if (Status.OK != status) {
            app.UIHandler.sendMessage(app.UIHandler.obtainMessage(
                    Colaboration.TO_SCREEN,
                    "Error en RegisterSignalHandler"));
            app.finish();
            return;
        }
        status = mBus.findAdvertisedName(NAME_PREFIX);
        if (Status.OK != status) {
            app.UIHandler.sendMessage(app.UIHandler.obtainMessage(
                    Colaboration.TO_SCREEN, "Error findAdvertise"));
            app.finish();
            return;
        }
        break;
```

Código 11. Fragmento de método handleMessage de la clase AlljoynHandler.

5.2.4.9 Módulo colaborativo: Comunicación entre dispositivos.

Clase: `AlljoynHandler.`

La comunicación de los dispositivos conectados al bus y registrados como emisores (`SignalEmitter`) se da a través de señales (`BusSignals`). Los dispositivos se registran para recibir señales por el método `registerSignalHandlers.` Cuando un dispositivo emite una señal al bus, la forma en que otros dispositivos la reconocen, en el módulo colaborativo, es a través del método *Action* asociado a la señal del mismo nombre (definida en la interfaz `PTInterface`). La anotación `@BusSignalHandler`, los atributos `iface` y `signal` y los parámetros de entrada brindan la información necesaria para marcar el método que será activado al recibir la señal del bus, ver Código 12.

```
@BusSignalHandler(iface = "com.jimdans.pt.service", signal = "Action")
public void Action(int... action) {
    app.Action(action);
}
```

Código 12. Método Action que será ejecutado al recibir la señal Action del bus.

En el Apéndice B se encuentra otro ejemplo de implementación.

El código completo de la aplicación *ShareDraw* se encuentra comentado; también, se anexa el *javadoc*, el cual documenta cada clase y sus métodos, permitiendo así analizar la estructura de la aplicación; además, se proporcionan dos manuales:

- Manual técnico: contiene la información tanto del hardware, como del software necesarios para implementar la aplicación; también contiene un procedimiento conciso de cómo instalar el software de desarrollo; así como los pasos a seguir para instalar la aplicación *ShareDraw* en un dispositivo móvil.

- Manual de Usuario: contiene información básica para hacer uso de la aplicación, así como su propósito y una explicación de la funcionalidad de cada componente de la aplicación *ShareDraw*.

6 Conclusiones y Perspectivas del Proyecto

La realización de este proyecto permitió adentrarnos al uso de nuevas tecnologías como son *Android* y *Alljoyn*, las cuales están en auge y continuo desarrollo. El aprendizaje obtenido ha sido de gran importancia, porque nos permitió: conocer nuevos paradigmas de desarrollo de aplicaciones, integrar distintos conceptos y tecnologías (*XML*, *Java*, *Android*, *Alljoyn*) en un solo proyecto; probar y experimentar el uso de distintas técnicas de programación; así como realizar un análisis y diseño puntual para el desarrollo del sistema.

La complejidad de este proyecto radicó en la investigación y experimentación de cómo funcionan tanto una aplicación en *Android*, como el funcionamiento e implementación de las clases e interfaces del *framework Alljoyn*, así como la integración de estas dos tecnologías para su uso en dispositivos móviles.

A continuación se describen los objetivos planteados en este proyecto; el grado de cumplimiento para cada objetivo; así como aquellos objetivos no propuestos y alcanzados.

6.1 Objetivos Propuestos Alcanzados

Objetivo: **Diseñar e implementar módulo de presentación que incluya un espacio privado y uno público.**
Se creó un módulo que genera una vista gráfica de los espacios público y privado de ShareDraw. Esto se logró a partir del empleo de principios de diseño gráfico que proporciona tanto *Android*, así como otros dos autores, Keim y Donald Norman. Los principios en los que nos basamos fueron: 1.-Simplificar la vida del usuario (*AndroidDesign, [13]*), 2.- representar con mínima entropía visual un conjunto de datos a un usuario final (Keim,[14]), 3.-*la* visibilidad de una interfaz deberá permitir que el usuario pueda encontrar las distintas funciones que se pueden realizar en la aplicación (Donald Norman, [15]).

Algunas consideraciones para la vista gráfica de *ShareDraw* que requirieron tiempo de aprendizaje, diseño e implementación fueron:

Android emplea una clase llamada `Activity`, la cual permite definir las vistas (`View`) gráficas de cualquier aplicación; siendo `Activity` el único medio para hacerlo. Por cada objeto `Activity` que se cree, se debe manejar su ciclo de vida, para definir, en el ámbito de la aplicación, si el objeto se debe interrumpir o reiniciar o inicializar.

La diversidad de dispositivos en cuanto al tamaño y la resolución de las pantallas, implicó implementar para cada espacio de dibujo una vista (View), por lo tanto nuestro objeto `ManejadorEspacios` (tipo Activity) gestiona un espacio a la vez.

La gestión de cambio entre espacios implicó implementar código que hiciera uso de un método de Android que permitiese hacer persistente la información del marco de dibujo (`DrawFrame`) en el momento de realizar una transición entre un espacio y otro. En Android un FrameLayout (`DrawFrame`) se extiende de `ViewGroup`, el cual permite agregarle sub-vistas (`View`) a partir de su método `addView`; sin embargo no lo muestra de manera inmediata en la pantalla, para ello hace uso del método `invalidate`, el cual fuerza al objeto `ViewGroup` a mostrar la vista más actualizada. Para hacer persistente la información de cada espacio de dibujo en ShareDraw, se hizo uso del método `invalidate` en la clase `EspacioPrivado` y

EspacioPúblico, debido a que es en estas clases donde los marcos de dibujo (DrawFrame) se crean y se agregan mediante *addView* a un *FrameLayout* general llamado *mainFrame*. Cuando los objetos *DrawFrame* han sido agregados como una vista más al *mainFrame* se llama a su método invalidate; el cual actualizará los marcos de dibujo de los espacios, cada vez que exista una transición entre espacios.

Objetivo: **Diseñar e Implementar un módulo para gestionar los objetos de los espacios privado y público.**
Se creó un módulo para gestionar los objetos de los espacios privado y público, a partir de un modelo conceptual de los objetos gráficos: círculo, rectángulo, línea, polígono y texto. La implementación de los objetos gráficos y su gestión se llevó a cabo a partir del paquete *graphics* y *widget* pertenecientes a *Android*.

En la implementación de estos objetos se utilizó el concepto de herencia, porque los distintos tipos de objeto comparten atributos y métodos. Para los objetos tipo polígono, fue necesario implementar un algoritmo para dibujarlo, debido a que las bibliotecas de *Android* no contenían un método para la creación de un polígono de n lados. También se creó un algoritmo para modificar el tamaño de los objetos gráficos.

Objetivo: **Diseñar un módulo para la transición entre los espacios privado y público.**

Se creó un módulo para la transición entre los espacios privado y público. Este módulo se logró diseñar de manera más efectiva en una tercera iteración en el desarrollo del sistema, debido a que hace uso de los módulos de gestión de objetos y de presentación.

Este módulo colabora más estrechamente con el módulo de presentación, ya que es necesario manipular tanto los objetos gráficos como el marco de dibujo que los contiene. Es importante mencionar la colaboración de estos dos módulos hace posible la persistencia de los datos en los marcos de dibujo de cada espacio, en el momento que se realice un cambio entre los espacios privado y público.

Objetivo: **Elaboración de la Documentación necesaria para el proyecto. Manual Conciso de Ayuda para el uso de la aplicación**
Se realizaron todos los documentos necesarios para el diseño e implementación del sistema ShareDraw, así como los manuales correspondientes.
Como punto extra, se realizó un pequeño manual que se encuentra dentro de la aplicación, para informarle al usuario sobre: 1) el propósito de la aplicación y 2) la descripción de cómo funciona cada uno de los botones.

6.2 Objetivos No Propuestos Alcanzados

Objetivo: **Adaptabilidad de la interfaz gráfica a partir del tamaño y resolución de la pantalla del dispositivo.**
Esta funcionalidad se realizó pensando en la diversidad de dispositivos que cuentan con un sistema Android. El rango de tamaño de pantallas en estos dispositivos va de 2.7 a 10 y 11 pulgadas. Además, las resoluciones de las pantallas no dependen de su tamaño, algunas pantallas de tamaño pequeño puede contar con una resolución de alta definición. Por lo tanto, fue necesario llamar, en el marco de dibujo, a los métodos de la clase `WindowManager`; estos métodos permiten obtener las dimensiones y resoluciones de los dispositivos; de tal manera que, antes de iniciar la aplicación ShareDraw en cualquier dispositivo móvil, se llamarán a estos métodos para manipular y presentar a los elementos gráficos adecuadamente.

Objetivo: **Funcionalidad de detección y manejo de eventos multitáctiles.**
Esta funcionalidad fue implementada a través de métodos de la clase DrawFrame; como una opción para editar el tamaño de los objetos gráficos; lo cual es un gesto común e intuitivo para los usuarios, en el empleo de aplicaciones en dispositivos móviles. Por tanto, implementar esta funcionalidad fue importante porque permite crear una interacción más dinámica entre el usuario y los objetos gráficos.

Objetivo: **Funcionalidad para proporcionar movimiento de un ObjetoGrafico en tiempo real entre distintos dispositivos.**

Esta funcionalidad fue implementada en base al modelo lógico de *Alljoyn* (presentado en la sección de arquitectura de software). La *framework Alljoyn* se encarga de manejar a nivel de capa de red aquellos aspectos que conlleva implementar un sistema distribuido como es: concurrencia, detección de fallos, tolerancias de fallos, recuperación frente a fallos, transparencia, entre otros. Por tanto, al manejar estos aspectos *Alljoyn* permite que los datos transmitidos por el sistema distribuido implementado de la aplicación *ShareDraw* fluyan de manera continua en tiempo real.

6.3 Objetivos Parcialmente Alcanzados

Objetivo: **Diseñar e Implementar un módulo de comunicación entre dispositivos móviles bajo una red *P2P***

Se creó un módulo que permitiera la comunicación entre dispositivos móviles, tanto para una red peer-to-peer como para una red cliente-servidor mediante el uso del framework Alljoyn. El

desarrollo de este módulo fue un proceso lento, debido a su complejidad. A continuación se describen las etapas principales de su implementación:

1) Instalación del *framework* en base a requerimientos estrictos, como son: utilizar una versión específica de java (*Sun Java* 5, difícil de encontrar en repositorios); cambiar la versión del sistema operativo condicionado por la versión de java; restringir la versión del API de Android (API nivel 8, Android versión 2.2) a utilizar para implementar *ShareDraw*, entre otros.
2) Experimentación y realización de pruebas sobre el *framework* de *Alljoyn* teniendo como objetivo la viabilidad y dificultad de implementar una red *p2p* en la aplicación. En esta etapa se presentaron las primeras dificultades debido a la fase inicial de desarrollo en la que se encontraba *Alljoyn*, ya que la existencia de documentación y ejemplos en los cuales nosotros pudiéramos comprender el funcionamiento y lógica de *Alljoyn* era reducida.
3) Se hicieron distintos diseños de cómo este módulo colaborativo podría funcionar eficientemente y ser integrado a la aplicación *ShareDraw*. Dichos diseños fueron implementados presentando diversas problemáticas, como fueron: falta de comunicación entre dispositivos a pesar de que se logró que se conectarán en red, el canal de comunicación no reconocía las señales; la trasmisión de las señales no se lograba (necesarias para implementar un modelo de comunicación *peer-to-peer*), falla de detección de cambios en el bus de comunicación.
4) Implementación del módulo colaborativo sin ser integrado a la aplicación *ShareDraw*, como resultado de las problemáticas descritas anteriormente, así como del tiempo de desarrollo para este proyecto. Sin embargo, se logró implementar este módulo bajo dos tipos de arquitectura de red: Cliente – Servidor, así como *peer-to- peer*. Este módulo en sus dos versiones (*peer-to- peer*, Cliente – Servidor) ofrece la posibilidad de editar una figura geométrica en su color, tamaño o posición transmitiendo los cambios en tiempo real a los demás dispositivos que se encuentren conectados a la red. Para lograr el funcionamiento del módulo en base a la arquitectura *peer-to-peer* fue necesario construir un módulo pequeño de registro de la sesión a la que se unirían los *peer's*, debido a que una vez que los p*eer*'s se conectaban a la red no se encontraban para colaborar; cabe mencionar que este mismo procedimiento funciona en la versión del módulo colaborativo Cliente-Servidor. Por tanto, para la versión *peer-to-peer*, antes de iniciar el modo colaborativo se tendrá que registrar o unir a una sesión el usuario (*peer*), en dado caso de que se encuentre un error en el registro de la sesión no se podrá iniciar el modo colaborativo a pesar de que se encuentren conectados los *peer's* a la red y al servicio de *Alljoyn*.

6.4 Objetivos a futuro

Finalmente, como trabajo futuro de este proyecto, proponemos algunas funcionalidades por desarrollar:

- Integrar el módulo de colaboración a *ShareDraw*, implementado en este proyecto.
- Implementar nuevas acciones a realizar en la aplicación, como son: dibujar a mano alzada, insertar imágenes, girar los objetos gráficos mediante multitouch, implementar un chat, etc.
- Implementar la aplicación ShareDraw en computadoras personales mediante el *framework* de Alljoyn.
- Implementar un módulo de colaboración vía *web* para la aplicación ShareDraw apoyando así el trabajo a distancia. Convirtiendo a ShareDraw en una aplicación *web*.

Sobre el último punto, es importante mencionar que actualmente existen diversas aplicaciones que cuentan con la posibilidad de colaborar a distancia; entre estas aplicaciones se encuentran: *Prezi*, *Google Docs* mencionadas en los antecedentes de este documento. *Prezi* ofrece un espacio de trabajo donde los usuarios pueden editar texto y cada uno de sus usuarios observan los avances de los demás, mientras que *GoogleDocs* presenta un editor de texto donde de igual forma los usuarios editan y colaboran en la creación de los documentos, sin la necesidad de estar conectados en la misma red.

Apéndice A

Escenario de Uso: GO-1: *Crear Objeto*

Actor Primario: Usuario

Actor Secundario: -

Disparador: El actor primario desea crear un **objeto**, ya sea una **figura** o **texto**, en el espacio de trabajo donde se encuentre.

Precondición: El actor primario se encuentra en un espacio de trabajo (privado o público).

Poscondición: La aplicación crea el objeto y lo muestra en el espacio.

Flujo Principal:

1. El actor primario oprime tecla o ícono menú, que contiene cualquier dispositivo con Android.
2. El actor primario selecciona, en el menú principal, la opción de Crear Figura[18].
3. La aplicación muestra un diálogo con las siguientes figuras a elegir: **línea, círculo**, **triángulo**, **cuadrado** o **pentágono** o signo más (polígonos), la cual al ser seleccionada mostrará una lista con los número de **lados** posibles a elegir para la figura poligonal (6, 7, 8,9,10,11,12).
4. El actor primario selecciona la figura o número de lado deseado.
5. El diálogo desaparece y aparecerá un mensaje "Toca la pantalla".
6. El usuario tocará la pantalla. La figura se creará y mostrará en la posición tocada por el usuario y en el **espacio** donde se encuentre el actor primario.

Flujo Alternativo:

1. El actor primario selecciona la opción de crear texto[19].

   1.1 La aplicación mostrará un diálogo con un cuadro de texto sobre el espacio de trabajo, de igual forma se mostrará un teclado virtual.

   1.2 El actor primario introducirá el texto deseado y deberá seleccionar "enter" al finalizar el texto.

   1.3 El teclado virtual se esconderá y aparecerá un mensaje "Toca la pantalla".

   1.4 El actor primario tocará la pantalla. El **texto** escrito se mostrará en la posición tocada por el usuario y en el espacio donde se encuentre el actor primario.

Requerimientos no funcionales:

En caso que el usuario se encuentre en el espacio público gestionando un objeto, será necesario comunicarse con el servicio de Alljoyn, para la actualización de los cambios realizados.

En caso de que la aplicación tardara en mostrar el objeto, se deberá retroalimentar al actor primario con una barra de progreso del procedimiento.

[18] Botón con tres figuras geométricas.
[19] Botón con letra A.

Diálogo de Ayuda para el Usuario

Figura A.1.

Diálogo para la elección de Color

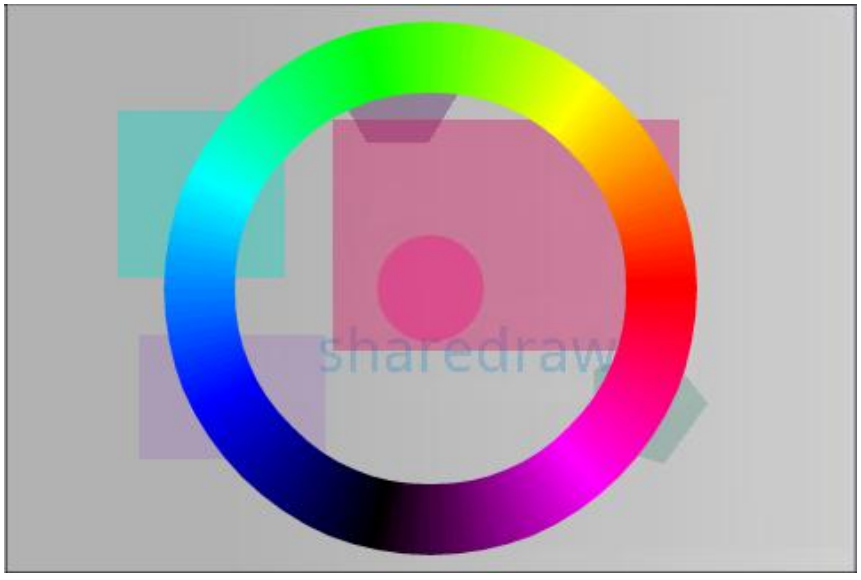


Figura A.2.

Apéndice B

Ejemplo de un Archivo XML, el cual permite que cada uno de los botones, de la aplicación, cambien a otra perspectiva retroalimentando al usuario.

trashl.xml *textoeditl.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android" >
    <item
        android:state_pressed="true"
        android:state_enabled="true"
        android:drawable="@drawable/textoeditshadow" />
    <item
        android:state_enabled="true"
        android:drawable="@drawable/textoedit" />

</selector>
```

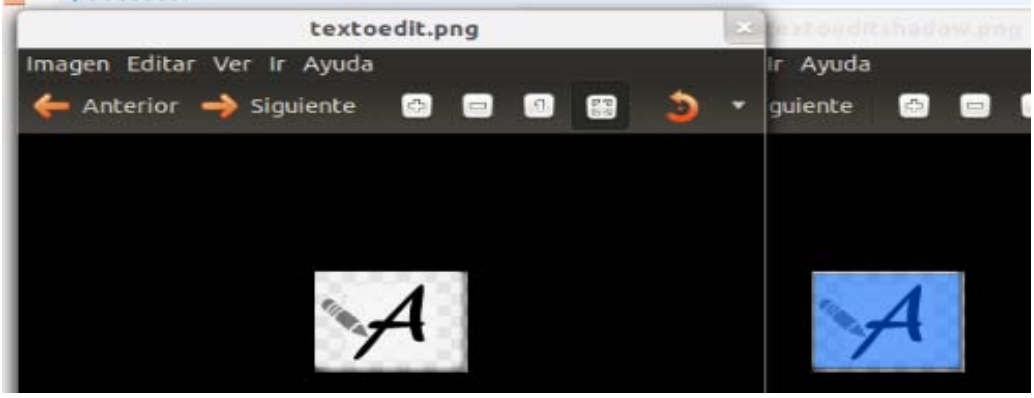


Figura B.1

Ejemplo de dos archivos xml que permiten animar el cambio entre una vista del `EspacioPrivado` *y el* `EspacioPúblico` *o viceversa.*

push_up_out.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2007 The Android Open Source Project

<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate android:fromYDelta="0" android:toYDelta="-100%p" android:duration="300"/>
    <alpha android:fromAlpha="1.0" android:toAlpha="0.0" android:duration="50" />
</set>
```

Design Source

push_up_in.xml

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Copyright (C) 2007 The Android Open Source Project

<set xmlns:android="http://schemas.android.com/apk/res/android">
    <translate android:fromYDelta="100%p" android:toYDelta="0" android:duration="300"/>
    <alpha android:fromAlpha="0.0" android:toAlpha="1.0" android:duration="50" />
</set>
```

Figura B.2

Interfaz Gráfica de desarrollo de Layouts

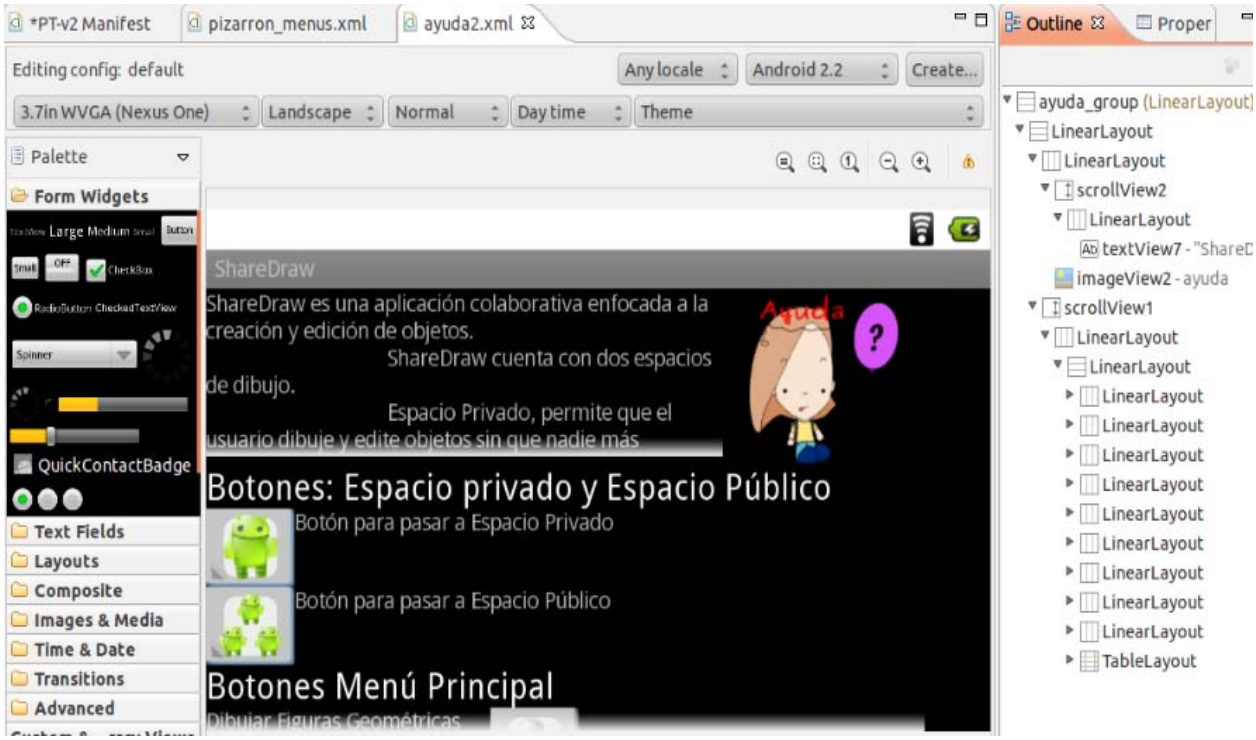


Figura B.3.

Pantalla Principal de Aplicación ShareDraw.

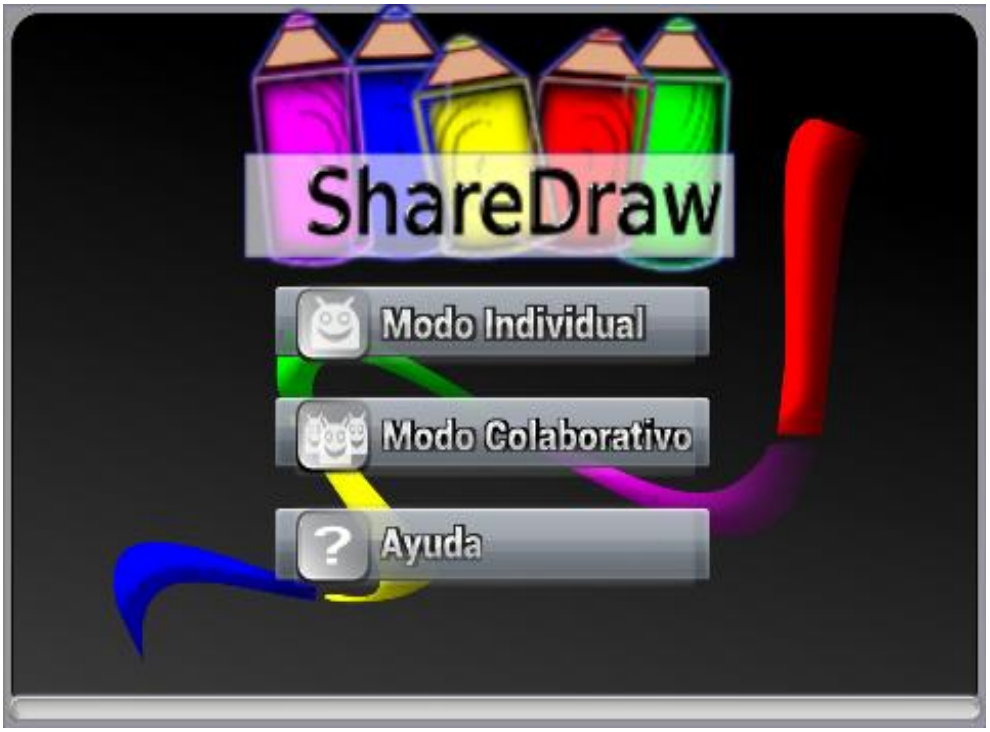


Figura B.4.

Pantalla modo colaborativo, Espacio Público.

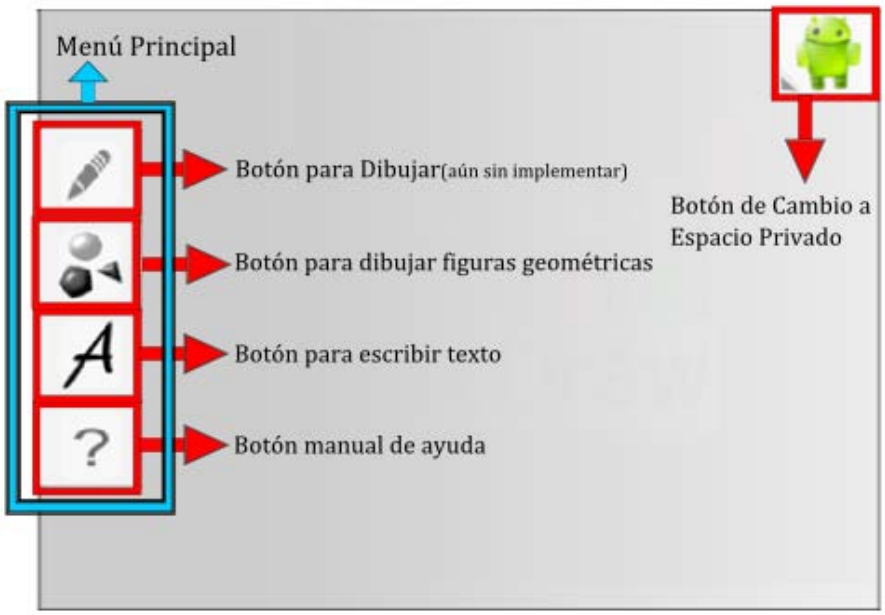


Figura B.5.

Cambiando Tamaño: Aumentando.

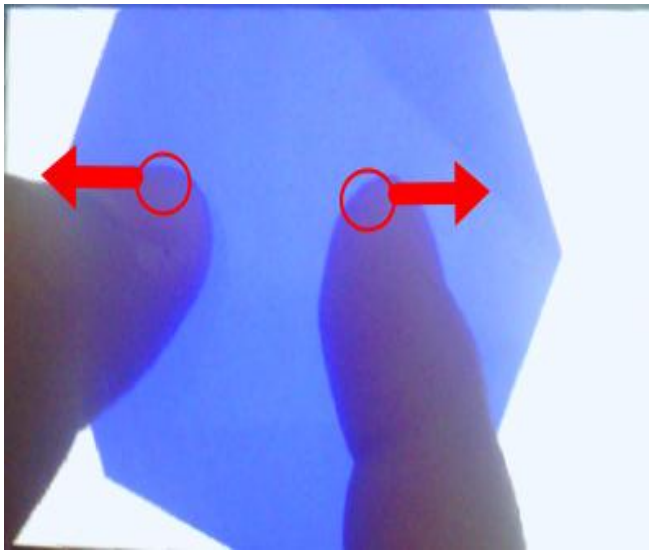


Figura B.6.

Cambiando Tamaño: Disminuyendo.

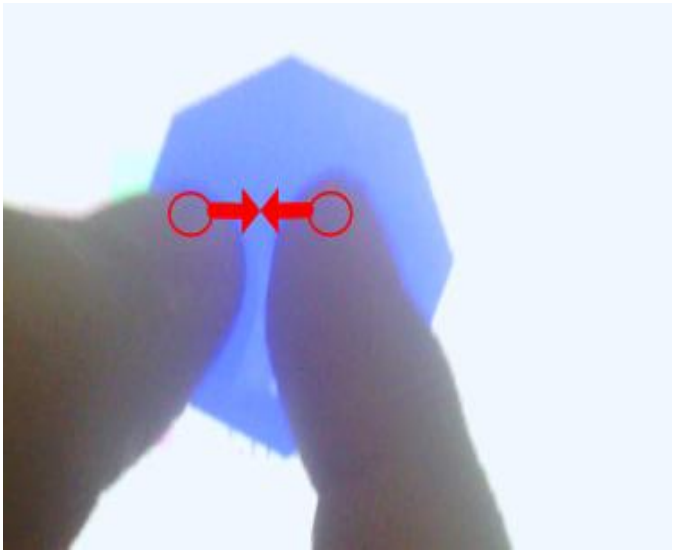


Figura B.7.

Ejemplo de Implementación: Editar Texto

Clase: `ShareDrawListener`

En su método Select existe la opción en `EDITAR_TEXTO`, la cual mostrará un diálogo. Como ya se había visto

```
LayoutInflater factory = (LayoutInflater) mContext
        .getSystemService(Activity.LAYOUT_INFLATER_SERVICE);
// carga layout para editar texto
final View crearTexto = factory.inflate(R.layout.crear_texto, null);
AlertDialog.Builder builder = new AlertDialog.Builder(
        v.getContext());
final EditText editText = (EditText) crearTexto
        .findViewById(R.id.cajaCreaTexto);
```

Una vez obtenida la vista se hace una instancia de la caja de texto del `layout`. Este es necesario para poder asignarle el mensaje que contiene el objeto `Texto` antes de ser editado. Como se hace en la primera línea de código siguiente.

```
editText.setText(frame.regresaTextoActual());
builder.setTitle("Escribe el Texto");
builder.setIcon(R.drawable.ic_launcher);
builder.setView(crearTexto);
builder.setPositiveButton("Ok",
        new DialogInterface.OnClickListener() {

            public void onClick(DialogInterface dialog, int which) {

                //cambiala cadena de objeto Texto por la cadena editada.
                frame.cambiarTextoActual(editText.getText() + "");
            }
        });
```

Una vez que el usuario toque el botón *Ok*, se asignará el nuevo texto escrito en la caja de texto al objeto Texto que había sido seleccionado. En las figuras 13, 14, y 15 se muestra los pasos de la edición de Texto.

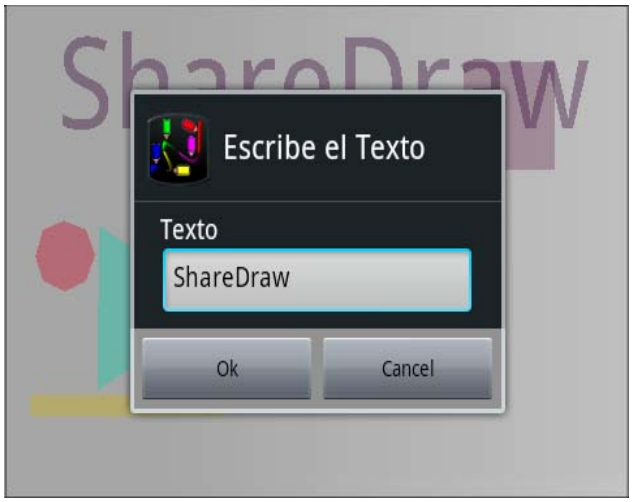


Figura B.8. Caja de Texto con texto actual.

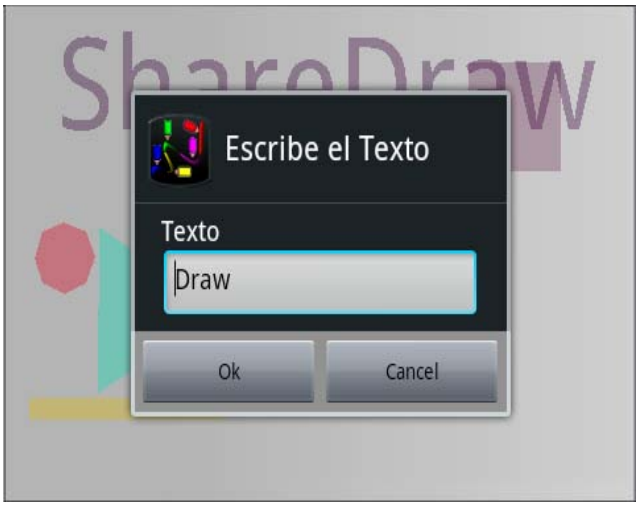


Figura B.9. Caja de texto con Texto Editado.

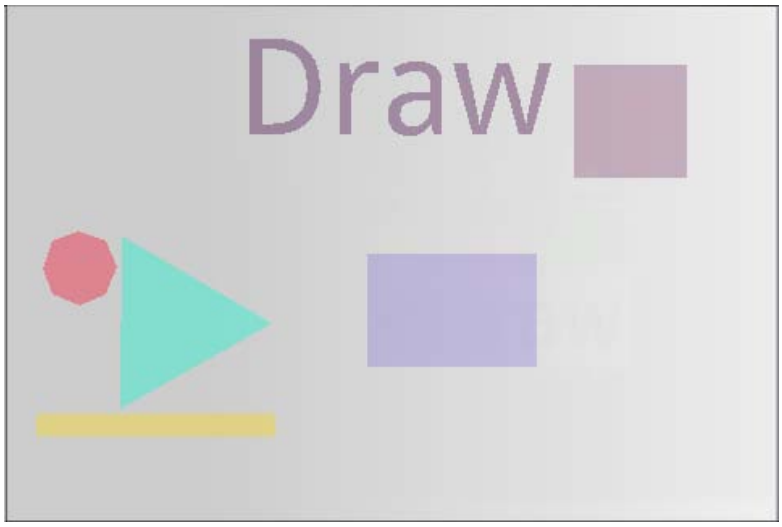


Figura B.10. Resultado Final de la Edición de Texto.

Glosario

Aplicación colaborativa	Referente a una aplicación que es ejecutada en varios dispositivos para trabajar en equipo
Groupware	Software que promete incrementar la productividad, ofreciendo a los usuarios una interfaz común con la que tienen acceso a una variedad de programas para trabajar de distintas maneras
Tareas Robustas	Tareas que gastan recursos considerables como memoria, procesamiento de operaciones, espacio de almacenamiento, interacción con otras aplicaciones como bases de datos y conexiones a redes inalámbricas.
P2P	Acrónimo de peer-to-peer un sistema distribuido de comunicación donde cada nodo puede compartir sus recursos solicitando y ofreciendo servicios dinámicamente
Portabilidad	Característica que posee un software para ejecutarse en distintas plataformas.
Vista gráfica	Referente a lo que se muestra al usuario en la pantalla del dispositivo.
Diseño Estético	Interfaz agradable basada en principios de diseño
APK	Un archivo con extensión .apk es un paquete para el sistema operativo Android. Este formato es una variante del formato JAR de Java y se usa para distribuir e instalar componentes empaquetados para la plataforma Android para smartphones y tablets.
Resolución	Formalmente resolución de pantalla, es el numero de pixeles que será mostrado en la pantalla
Retroalimentación	Se refiere a mensajes de vuelta cuando el usuario introduce una entrada para mantener al usuario al tanto de lo que sucede en la aplicación.

Gestos	Movimiento de la mano que expresa algo, en nuestro ámbito se refiere a los ademanes que alteran el comportamiento de algún objeto en Android. Ejemplo: el conocido "pellizco" para agrandar o hacer más pequeño un texto.
Espacio público/privado	Referente a las pantallas que se muestran al usuario y a los menús información y vistas que ofrecen dichas pantallas. El espacio público es aquel en el que el usuario comparte información a través de la red y el espacio privado es donde él puede realizar sus dibujos y únicamente su dispositivo puede ver lo que hace (hasta que lo comparta).
MultiTouch	La tecnología multitouch consiste en una pantalla táctil que reconoce simultáneamente múltiples puntos de contacto, así como el software asociado a esta que permite interpretar dichas interacciones simultáneas.

Bibliografía

[1] G. Sánchez Morales, "Redistribución semi-plástica mixta: el caso de estudio de un pizarrón compartido", Tesis de Maestría, Departamento de Computación, CINVESTAV IPN, febrero 2009.

[2] M. Gargenta, *Learning Android*, Primera Edición, California, O´Reilly Media, 2011.

[3] M. Papadopouli and H. Schulzrinne, *Peer-to-Peer Computing for Mobile Networks Information discovery and Dissemination*, NewYork, Springer, 2009.

[4] A. S. Tanenbaum, *Redes de Computadoras*, Cuarta Edición, Estado de México, MX, Pearson Prentice Hall, 2003.

[5] A. Morales García, "Transición de objetos compartidos entre espacios privado y público en un pizarrón compartido", Proyecto Terminal, Ingeniería en Computación, Unidad Azcapotzalco, UAM, D.F., México, julio 2010.

[6] J. Hernández Cruz, "Remodelación plástica de una barra de colaboradores para un espacio de trabajo dentro de un pizarrón compartido", Proyecto Terminal, Ingeniería en Computación, Unidad Azcapotzalco, UAM, D.F., México, junio 2010.

[7] Silberschatz, et. al., *Operating System Concepts*, Séptima Edición, Massachussets, John Wiley & Sons, 2005.

[8] H. Juárez Cerón, "Remodelación plástica de la barra de herramientas de un pizarrón compartido", Proyecto Terminal, Ingeniería en Computación, Unidad Azcapotzalco, UAM, D.F., México, junio 2010.

[9] Prezi inc, (2011,10, 24), *Prezi*, [En línea] Disponible: http://prezi.com .

[10] Google, (2011, 10, 26), *Google docs: Comparte y colabora en tiempo real*, [En línea], Disponible: http://www.google.com/google-d-s/intl/es/tour2.html

[11] Android Google, (2011, 10, 26), *Android Market*, [En línea], Disponible: https://market.android.com/details?id=com.google.android.apps.docs&hles.

[12] C. Larman, *UML y Patrones Introducción al análisis y diseño orientado a objetos*, Prentice Hall, noviembre 2004.

[13] Android Google, (2012, 01, 10), Android Design, [en línea], Disponible: http://developer.android.com/design/get-started/principles.html

[14] NoSoloUsabilidad, (2012, 04, 08), Hacia una taxonomía de investigación entre Visualización de información y Diseño, [en línea], Disponible: http://www.nosolousabilidad.com/articulos/taxonomia_visualizacion.htm

[15] D. Norman, *La psicología de los objetos cotidianos*, primera edición, New York, Nerea, 1988.