Universidad de Chile
Facultad de Ciencias Físicas y Matemáticas
Departamento de Ciencias de la Computación

Memoria para optar al título de ingeniero civil computación

NetSketcher: Plataforma para el desarrollo de aplicaciones colaborativas y gestuales sobre redes ad-hoc.

Gabriel Alonso Peña Pascual

Profesor Guía:
Nelson Antranig Baloian Tataryan

Miembros de la comisión:

Gustavo Zurita Alarcón
Andrés Farías Riquelme

Santiago, Chile
Septiembre 2010

Resumen de la Memoria para optar
al título de ingeniero civil en
computación
Por: Gabriel A. Peña Pascual
Fecha: 02/09/2010
Prof. Guía: Sr. Nelson Baloian T.

NetSketcher: Plataforma para el desarrollo de aplicaciones colaborativas y gestuales sobre redes ad-hoc.

Los equipos de trabajo en cualquier escenario requieren mantener una estrecha comunicación para lograr un objetivo conjunto y desde un principio la ciencia de la computación ha estado al servicio de las personas entregando herramientas para facilitar la colaboración.

En las últimas décadas la aparición de los equipos móviles de forma masiva y el aumento de potencia, ofrecen una nueva área de investigación para generar herramientas colaborativas.

El presente trabajo propone una plataforma que ofrezca soporte para el desarrollo de aplicaciones colaborativas orientada al trabajo en una red ad-hoc. El objetivo de la plataforma es simplificar la creación de aplicaciones colaborativas que permitan continuar con la investigación en esta área y de paso recabar las mejores propiedades de trabajos anteriores.

La plataforma que se desarrolló consiste en 3 capas: comunicación, sesión y espacio de trabajo, cada una dependiente de la anterior. El sistema conjunto permite realizar aplicaciones colaborativas con interfaz usuaria gestual. Además para probar la plataforma se implementó un conjunto de aplicaciones y herramientas para el apoyo de testeo.

Como prueba de conceptos se desarrollo una aplicación para diagramar procesos de negocio, esta aplicación, llamada FlowChart Maker, está orientada a la planificación de proyectos en las primeras etapas del diseño.

Del trabajo se puede concluir que la plataforma cumple con los objetivos de comunicación, usabilidad, interfaz usuaria y mantenimiento.
Y como trabajo a futuro se plantea extender la plataforma permitiendo sesiones asíncronas, realizar estudios de nuevas formas para generar gestos y reforzar la comunicación a través de Bluetooth

Agradecimientos

Gracias a mi profesor guía Nelson Baloian, por la confianza que puso en mí y por el apoyo que siempre me ofreció a través del proyecto.

Gracias al profesor Gustavo Zurita por sus consejos que me permitieron hacer un mejor trabajo.

Gracias también a todas las personas que se involucraron con este proyecto y me ayudaron a darle forma a este trabajo.

Y por último a mi familia, que siempre han creído en mí y me han acompañado durante todo este viaje, muchas gracias.

Tabla de Contenidos

1 Introducción

En las últimas décadas hemos visto una revolución en la miniaturización, disminución de costos e incorporación de dispositivos tecnológicos en nuestra vida diaria [2]. Múltiples equipos móviles han cautivado el interés de la sociedad, logrando penetrar ampliamente en la vida de las personas. Ya sea por diversión o por trabajo, los equipos móviles, provistos de conexión en todo momento, nos mantienen cerca, comunicados, relacionados unos con otros de maneras que todavía estamos descubriendo.

Esta nueva era de conectividad inalámbrica hace posible explorar el trabajo colaborativo computacional (CSCW) vinculado con la computación ubicua, permitiendo utilizar a la computación como un medio casi imperceptible de colaboración entre personas.

El término computación para el trabajo colaborativo (Computer Supported Collaborative Work o CSCW) fue acuñado por Grief y Cashman en 1986 y se puede definir como un sistema computacional que asiste a un grupo de personas dedicadas a una tarea común [6].

En el caso de Computación ubicua, según Mark Weiser, se define como la capacidad de embeber la tecnología en nuestro entorno y hacerla prácticamente invisible [2].

La combinación de estos 2 conceptos genera un área de investigación no definida aun, sin embargo está en creciente aumento la variedad de aplicaciones que se están desarrollando en esta materia, pues es posible brindar apoyo computacional en situaciones que antes eran imposibles. Por ejemplo, se puede brindar apoyo a personas que van viajando en un tren o están inspeccionando una obra en construcción [7].

Aun que desde hace mucho tiempo existe una variedad de herramientas que entregan soporte para el trabajo colaborativo: Coast [14, 10], DistView [14], Habanero [17], GroupKit [14, 10]. Sin embargo son inmediatamente descartadas cuando se trata de computación móvil. Para realizar una aplicación móvil y colaborativa nos encontramos con un problema: debemos empezar a desarrollar desde muy bajo nivel para alcanzar nuestro objetivo.

Las aplicaciones colaborativas móviles plantean problemas distintos a las aplicaciones fijas: dos de los mayores problemas para desarrollar Groupware síncrono son la replicación de datos y mantener múltiples sesiones de conexión[5], esto sobre dispositivos con limitada capacidad y redes no siempre disponibles , hace

que muchas plataformas de sistemas distribuidos pensadas para la computación fija no sean aplicables al área móvil, por lo que se han desarrollado, en el último tiempo, plataformas exclusivas para apoyar escenarios de computación móvil.

Existen variadas propuestas, pero la mayoría no ha obtenido muy buenos resultados, un caso conocido es JXTA que ha debido pasar muchas dificultades y que aun hoy en día continúa con problemas para entregar el soporte que propone [20].

Podemos evidenciar que todas estas plataformas para ambientes colaborativas tan solo adaptan paradigmas clásicos de computación, por ejemplo en la interacción con el usuario, manteniendo los mismos conceptos de las primeras GUIs (Xerox PARC en 1973), o también en la forma de interacción entre pares (sistemas centralizados, arquitectura cliente-servidor, etc.) [29].

Es por esto que se entrega una nueva propuesta de plataforma colaborativa, basada en conceptos de aplicaciones colaborativas desarrolladas en el Departamento de Computación de la Universidad de Chile en los últimos 3 años e incluyendo conceptos y estructuras que han probado ser indispensables en plataformas para redes ad-hoc.

En conclusión, se busca crear una nueva plataforma para aplicaciones colaborativa sobre redes ad-hoc, que entregue soporte a aplicaciones donde los usuarios puedan interactuar por medio de gestos y realizando dibujos sobre pantallas táctiles, con el fin de hacer más sencilla e intuitiva la interacción humano-computador.

1.1 Motivación

Hoy en día una plataforma confiable para apoyar el desarrollo de Groupware, cuentan con desarrollos de capas bien definidas, un buen manejo de objetos compartidos, control de usuarios y mantienen una independencia entre capas [3, 4,20].

Es necesario un buen diseño de estas capas para facilitar el desarrollo, la realización de testeos y mantención del código, de modo que la plataforma provea una modularización y documentación que faciliten el desarrollo sobre esta.

Actualmente en el Departamento de Ciencias de la Computación de la Universidad de Chile se cuenta con un sistema, llamado SmartDraw, de características colaborativas en redes ad-hoc e interfaz por medio de gestos. Sin embargo, a causa de su creciente expansión y al aumento de aplicaciones que se han desarrollado con

ella, SmartDraw mezcla en una sola capa muchas funcionalidades que la han vuelto difícil de manipular.

Debido a esto y que las plataformas más reconocidas hoy en día no entregan un soporte satisfactorio al área de computación colaborativa en redes ad-hoc, se hace necesario crear una nueva plataforma que reúna las distintas cualidades deseadas:

- Sincronización de información en una red descentralizada.
- Transparencia en comunicación para programadores y usuarios.
- Soporte de múltiples grupos en paralelo y políticas de control para objetos compartidos.
- Interfaz basada en gestos (ingreso de información por medio de escritura y dibujos a mano alzada) y utilizar elementos gráficos como medio de interacción colaborativa.
- Optimizar el uso de recursos.
- Disminuir tiempos de desarrollo, automatizar y simplificar las funcionalidades de la plataforma.
- Brindar un conjunto de elementos de soportes para el desarrollo sobre la plataforma.

Basado en la interacción humano-computador que cuenta SmartDraw e incluyendo el sistema de comunicación para redes peer-to-peer llamado High Level Manet (HLM) [18,19], se diseñará e implementará una plataforma que ofrezca un soporte completo para el desarrollo de aplicaciones colaborativa.

La plataforma, llamada NetSketcher, se desarrollará buscando aprovechar de la mejor manera los recursos de equipos móviles, reducir tiempos de desarrollos, facilitar las etapas de testeo, facilitar su extensibilidad y presentarla como una opción, simple y efectiva, para desarrollar aplicaciones colaborativas sobre redes ad-hoc.

1.2 Objetivos

El objetivo general de este trabajo de título es realizar una plataforma para implementar aplicaciones colaborativas en redes ad-hoc de una manera simple y efectiva.

Los objetivos específicos que se desea lograr en el trabajo de título son:

- Lograr un buen manejo de objetos que se crean y se comparten entre las aplicaciones durante el trabajo colaborativo, es decir, los que pertenecen a la sesión, a través de manejo diferenciado entre objetos compartidos y locales, entregando una interfaz que otorgue mayor control sobre los objetos a capas superiores, utilizando mensajes de control para entregar confiabilidad en la sincronización de una sesión y además estableciendo permisos y autorías de objetos
- Administrar grupos de usuarios, integrando control de acceso a usuarios a través de roles o permisos, permitiendo a usuarios participar de múltiples grupos a la vez y creando sistemas de invitación y petición de ingreso a grupos.
- Desarrollar una aplicación para validar el nuevo sistema desarrollado, esta aplicación permitirá realizar diagramas de procesos de forma colaborativa y utilizará todas las capas de la plataforma.
- Desarrollar una capa encargada de entregar soporte al realizar acciones sobre las aplicaciones colaborativas, que sirvan de base para posteriores aplicaciones que se realicen y que sea sencilla e intuitiva de usar (gestos).

1.3 Metodología

Para el plan de trabajo se utilizó un Modelo Incremental de 3 iteraciones, donde se dio principal atención al diseño e implementación de cada etapa. Además previo al desarrollo de la plataforma se realizó una etapa investigativa para estudiar la implementación de la administración de objetos sincronizados y recabar información de la aplicación SmartDraw (administración de objetos, modos de implementar aplicaciones, captura de eventos desde pantalla, etc.) con el fin de considerar los algoritmos que dieron mejor resultado y desechar los otros.

1.3.1 Primera Iteración

En la primera iteración se diseñó e implementó la capa de sesión, se estableció basada en diagramas la estructura de la capa y se procedió a implementar la capa de sesión.

En la capa de sesión se implementaron 4 módulos: el sistema de repositorio, el de sincronización de objetos, administración de sesiones en red y control de permisos de uso para usuarios sobre los objetos. Además se incorporó una interfaz para la capa de comunicación.

Luego se realizó una sencilla aplicación de testeo, que permite estudiar las funciones de creación y actualización de objetos dibujables, también tiene la cualidad de crear grupos. Gracias a esta aplicación se realizaron variadas pruebas, para asegurar el correcto funcionamiento de la capa de sesión y capa de comunicación.

1.3.2 Segunda Iteración

En la segunda interacción se realizó un diseño basado en lo investigado sobre SmartDraw, se implementó la capa de Trabajo, la cual cuenta con las interfaz para captura de eventos, los métodos de dibujo, las estructuras que asisten a la creación de aplicaciones (Mode-Gesture-Sketch) y un conjunto de elementos que dan soporte a la usabilidad de las aplicaciones. Además se implementó una aplicación de testeo llamada Drawing Board la cual incluye todos los gestos implementados.
.
Una vez terminada la implementación se procedió a ejecutar pruebas, realizando pruebas de usabilidad, red y funcionalidad de las distintas propiedades de la plataforma, además se incluyó la creación de una documentación de la API como apoyo y una guía para realizar aplicaciones sobre la plataforma NetSketcher.

1.3.3 Tercera Iteración

Luego, en la última etapa del trabajo de título, se realizó una aplicación para utilizar sesiones de red y una aplicación demo enfocada en trabajo colaborativo, también se continuó realizando pruebas de red, usabilidad y funcionalidad de la plataforma.

Al final se entrega un sistema integro con documentación, aplicación y demos.

2 Marco Conceptual

2.1 Estado del Arte

Para lograr realizar una plataforma con éxito es necesario analizar cuáles son las características fundamentales que debe poseer una plataforma con soporte colaborativo.

Las plataformas, al ser sistemas distribuidos pueden agruparse según su forma de compartir objetos en 3 tipos:

- **Arquitectura de Estado Centralizado:** Todas las operaciones de entrada o de salida se dirigen a un servidor, que también gestiona los datos compartidos. Los clientes sólo tienen la tarea de mostrar los datos compartido y enviar las solicitudes a un servidor central [9, 15].
- **Arquitectura de Estado Replicable:** La aplicación en sí y los datos compartidos se repiten. Ya que los datos compartidos solo se pueden acceder a nivel local por el usuario, se reduce el tiempo de respuesta comparado con arquitectura centralizada, aunque los algoritmos son más complejo [9,15, 10].
- **Arquitectura de Estado Hibrido:** Esta arquitectura se compone de las dos anteriores. Los datos compartidos pueden ser centralizados o replicarse [9].

Debido al tipo de hardware que se utiliza en este trabajo, se utilizará una arquitectura replicable que permita una red dinámica, donde todo nodo es un cliente y servidor a la vez, pero esta opción trae consecuencias, una de las más importantes es cómo administrar objetos compartidos sincronizadamente. La sincronización de objetos compartidos requiere crear un protocolo de comunicación que de seguridad a la hora de resolver problemas como colisiones de eventos simultáneos, es necesario que ofrezca confiabilidad en la ejecución de eventos en toda la red.

Un protocolo muy exitoso es el **Modelo de Secuenciamiento Central**, su facilidad de implementar y la variedad de implementaciones (terminal link, centralized data, distribute data) la hacen una opción muy interesante, en sí, el Modelo de Sequenciamiento Central se basa en un nodo central que maneja las transacciones en la red [13].

También existen protocolos no centralizados que eliminan el cuello de botella producto de un servidor central, se tiene el caso del **Modelo de Secuenciamiento Distribuido** que utiliza un sistema de bitácoras (log) para almacenar los eventos y

ejecutarlos en orden de prioridad, su implementación es bastante compleja y su uso de la red es ineficiente comparado con las demás opciones [1,13].

Por último, el **Modelo de Objetos Independientes e Inmutables**, un modelo no centralizado que corrige las fallas del Modelo Distribuido, este modelo controla el último estado del objeto y la prioridad que tiene el usuario para modificar un objeto, su implementación es sencilla y tiene un uso eficiente de la red, sin embargo presenta restricciones en el tipo de aplicaciones que se pueden desarrollar, ya que los eventos deben ser conmutativos, aditivos e independientes [13].

Varias de las plataformas y aplicaciones analizadas comparten cierto conjunto de capas, esencialmente 3, que describen la comunicación, administración de sesión y el control de eventos del usuario. A continuación una explicación de cada capa:

- **Capa de Comunicación:** dedicado a establecer conexión, envió y recepción de objetos a través de una red [3, 4, 11, 12].
- **Capa de Administración de Sesión:** capa dedicada a la administración de paquetes, a su persistencia en las sesiones colaborativas y la sincronización de los cambios que en ellos se produzcan [3, 4, 11,12, 16, 14].
- **Capa Espacio de Trabajo:** capa encargada del manejo de aplicaciones, intercepta los eventos del usuario y los envía a la Capa de Sesión [3, 4, 11,12].

No solo los objetos deben ser compartidos en la red, también los usuarios deben ser representados en la red, una forma simple para hacer esto es usando Roles.

Roles es un tipo de implementación que se usa cuando los usuarios tienen diversos derechos (o restricciones), según lo investigado se han encontrado dos tipos de políticas, Roles estáticos y dinámicos [16, 11].

- **Roles tradicionales o estáticos:** los roles son asignados tempranamente a los usuarios y rara vez son cambiados (estáticos), generalmente este tipo de roles se definen en función de sus miembros. En cuanto a sus desventajas, lo más importante es su falta de flexibilidad, ya que asignar el rol prematuramente, crea grupos de usuarios rígidos, además exige una participación explícita del usuario para configurar el rol [16].
- **Roles dinámicos:** roles asignado en tiempo de ejecución a medida que se realicen los eventos, en este caso el rol de un usuario es determinado por el “predicado” de la función que desea realizar. Este tipo de política a diferencia de la tradicional se define en base a los atributos de los usuarios [16].

Una plataforma con soporte colaborativo no solo debe ocuparse del control de flujo en la red, sino que debe permitir la visualización y organización de estos datos para cada usuario. Para mantener el conjunto de datos que se utilizan en una sesión se hace necesario un medio de almacenamiento y administración de los datos. Una buena solución es un repositorio, que en sí, son colecciones de pares llave-valor que representan los datos de una sesión colaborativa, los repositorios cuenta con varias características, entre las más comunes están [14, 16]:

- La habilidad de borrar y crear objetos que “viven” en la sesión.
- Vistas, que entregar subconjuntos de los datos almacenados.
- La meta-información de un objeto (fecha de creación, autor, tipo de objeto, etc.), que permiten crear políticas de acceso a cada objeto.

La información presentada aquí muestra muchas de las características que se buscan en una buena plataforma con soporte colaborativo, y el objetivo de este trabajo apunta a eso, entregar una implementación que satisfaga las necesidades de uso eficiente de red y un óptimo manejo de datos, buscando crear una plataforma actualizada y por sobre todo que entregue soporte a las aplicación colaborativas.

2.2 Trabajos Relacionados

Debido a que el área colaborativa es específica, se estudiaron variadas plataformas, frameworks y librerías que permitan la creación de una red colaborativa, considerando esto se encontraron los siguientes sistemas:

TOP (“Ten Object Platform”) [5, 14]: Plataforma que provee soporte al desarrollo de aplicaciones colaborativas en WEB, posee una arquitectura centralizada y permite aplicaciones tanto sincrónicas como asincrónicas, se basa en el patrón Broker para proveer sincronización y está implementado en JAVA y JavaScript.

COAST ("COoperative Application System Toolkit") [10]: Herramienta para el desarrollo de Groupware sincrónico de documentos, posee una arquitectura de replicado, incluye administración de sesiones, un sistema de control transaccional sobre los objetos compartidos y cabe destacar que la interacción con el usuario se hace utilizando un paradigma MVC.

PASIR (“Platform for Ad-hoc Sharing Information Resources”) [11]. Plataforma para compartir documentos, permite manejo de sesión y posee una arquitectura de replicación de objetos. Basado en la información encontrada la plataforma

actualmente soporta sesiones asincrónicas, pero se menciona que está en desarrollo la comunicación sincrónica.

Mobile Chedar (CHEap Distributed ARchitecture) [20,21]: Middleware que permite a equipos móviles crear una red peer-to-peer, utiliza un sistema centralizado con un PC que hace de puerta de enlace con internet y mantiene diversos tipos de recursos compartidos (datos, software y hardware).

Proem [20, 23,24]: Framework para el desarrollo de aplicaciones en redes peer-to-peer, su principal objetivo es proveer un desarrollo rápido de aplicaciones para redes ad-hoc. Proem está implementado en J2SE limitándolo a dispositivos más poderos, se ha intentando crear una versión en J2ME, pero aun no ha tenido éxito.

Groupkit [10, 14, 26,27]: Librería que provee una arquitectura síncrona de comunicación para aplicaciones multi-usuario, incluye políticas de control por capas y una interfaz gestual.

JXTA [20, 25,28]: Plataforma open-source, implementada en JAVA, para desarrollo de aplicaciones peer-to-peer, provee un set de protocolos y API multipropósito para comunicación entre computadores.

JXME [20,25]: Solución de JXTA para equipos móviles, implementado en J2ME, debido a las limitantes de J2ME utiliza un *"super-peer"*, llamado JXTA Relay, encargado de coordinar la red.

JMobiPeer [20,25]: Framework para redes peer-to-peer que provee administración de grupos, manejo y descubrimiento de nodos, tiene arquitectura replicante y pretende solucionar los problemas de JXME.

Peer2Me [20]: Framework para desarrollar aplicaciones peer-to-peer sobre equipos móviles, cuenta con un desarrollo modular implementado en J2ME y la comunicación se realiza vía Bluetooth.

Plataformas para redes ad-hoc

	PASIR	M. Chedar	Proem	JMobiPeer	Peer2Me	JXME
Sesiones de red	✓			✓	✓	✓
Repositorio colaborativo	✓					
Operativo en equipos de recursos reducidos	✓	✓			✓	✓
Interfaz gestual						
Arquitectura descentralizada	✓		✓	✓		
Operativo en redes ad-hoc	✓	✓	✓	✓	✓	✓
Colaboración síncrona		✓	✓	✓	✓	✓

Otras Plataformas Relevantes

	TOP	COAST	Groupkit	JXTA
Sesiones de red	✓	✓	✓	✓
Repositorio colaborativo	✓	✓		
Operativo en equipos de recursos reducidos		?	?	
Interfaz gestual			✓	
Arquitectura descentralizada		✓		
Operativo en redes ad-hoc				
Colaboración síncrona	✓	✓	✓	✓

De los 2 conjuntos de sistemas analizados se concluye que la mayoría no ofrece soporte para equipos móviles, aun cuando algunos fueron pensados para esto. La mayoría carece de interfaces gestuales, con especial atención podemos ver que los sistemas para redes ad-hoc no contemplan esta propiedad, estas dos características, al parecer, están relacionadas. Además se puede apreciar que casi ningún dispositivo considera la administración de datos colaborativamente, algo que un sistema que provee soporte para CSCW debe incluir.

En resumen y basado en el estudio de estos y otros sistemas se considerará con especial interés entregar estos 3 puntos: Repositorios colaborativos, Interfaz gestual y funcionamiento en equipos de recursos reducidos. Ya que son los puntos más deficientes en los sistemas estudiados.

2.3 Recursos utilizados

2.3.1 High Level Manet Protocol (HLMP)

Librería encargada de la comunicación en red de la plataforma a través del intercambio de mensajes. HLMP API está compuesto de 2 módulos, el *"Core"* y *"Plug-ins"*.

El *"Core"* implementa el mecanismo de soporte para los procesos de comunicación, intercambio de datos y procedimientos de interoperabilidad entre la API y el sistema operativo (S.O.) [19]

Por otro lado, el módulo de *"Plug-ins"* contiene especificaciones para la estructuración de protocolos de comunicación de groupware, los cuales utilizan los servicios provistos por el *"Core"*. Además incluye sencillas GUIs para la comunicación en redes ad-hoc [19].

Para este proyecto se utilizó el módulo de *"Core"*, el cual a su vez se divide en 3 sub-estructuras:

- **System Interoperability:** encargada de manejar las funciones delegadas al S.O., algunas de ellas son: la configuración del WLAN, administración del adaptador de la red wireless, configuración de la dirección IP y la detección de direcciones duplicadas [19].

- **Network Layer:** esta componente implementa los servicios de mensajería TCP y UDP en la MANET. Estos mensajes son validados y puestos en cola para ser utilizados en Communication Layer [19].

- **Communication Layer:** interfaz para manipular la API, esta componente también administra los servicios de HLMP, como: organización de mensajes, ruteo, packing y unpacking de mensajes [19].

El principal objetivo del Protocolo HLM es establecer una automatización en la creación de una Wi-Fi MANET y que los equipos puedan entrar a la red y colaborar [18].

Las redes de este tipo son dinámicas y la disposición de los nodos de la red cambian constantemente, es por esto que HLMP recalcula el camino de un mensaje en cada nodo por el cual es transmitido, para ello HLMP genera un gráfico de costos en cada nodo, actualizándolo constantemente con mensajes de tipo *"I'm alive"* [18].

El protocolo provee 4 tipos de mecanismos para envió de mensajes: Unicast, Multicast, SafeUnicast y SafeMulticast

Para trabajar en la Plataforma NetSketcher se utilizaron las opciones Safe, ya que este sistema asegura la recepción de un mensaje enviando hasta que exista confirmación por parte del receptor.

Basado en la calidad de la señal y el nivel de tráfico en los nodos conocidos en la red, se genera una matriz de costos, esta matriz es utilizada para asignar el costo para los caminos en la red. Cuando se intenta mandar un mensaje de tipo Safe se selecciona el mejor camino al vecino cercano y luego se utiliza comunicación TCP para enviarlo, después de la llegada de el mensaje a su destino, el receptor envía una confirmación de recepción del mensaje al emisor [18].

2.3.2 Dispositivos utilizados

Para realizar las pruebas se utilizaron PDAs, (Personal Digital Assistant), son computadores de mano que originalmente fueron diseñados como agendas electrónicas y cuentan con un sistema de reconocimiento de escritura, también se utilizaron Tablet PCs, computadoras que mezclan características de computador portátil y de PDA, en ellas se puede escribir a través de una pantalla táctil utilizando un stylus.

Algunos datos relevantes de los equipos utilizados:

Pocket PC Dell Axim X51v
Procesador: 624 MHz
Memoria Ram: 64 MB
Sistema Operativo: Microsoft Windows Mobile 5.0
Pantalla: 480x640 píxeles, 65536 colores

Pocket PC HP IPAQ hx2490b
Procesador: 520 MHz
Memoria Ram: 64 MB
Sistema Operativo: Microsoft Windows Mobile 5.0
Pantalla: 240x320 píxeles, 65536 colores

Smartphone Samsung Omnia II
Procesador: 800 MHz
Memoria RAM: 256 MB
Sistema Operativo: Microsoft Windows Mobile 6.5
Pantalla: 800x480 píxeles

Tablet PC HP Compact tc4400
Procesador: Intel® Core™ 2 Duo Processor T5600 (1.83 GHz)
Memoria RAM: 512 MB
Sistema Operativo: Microsoft Windows XP Professional (Service Pack 3)
Pantalla: 1024x768 píxeles

Notebook Acer Aspire 5520
Procesador: AMD Turion 64 X2 (1.9 GHz)
Memoria Ram: 2.048 MB
Sistema Operativo: Microsoft Windows XP Professional (Service Pack 3)
Pantalla: 1280x800 píxeles

Debido a diversos requerimientos de las dependencias, los equipos debían contar con S.O. específicos.

Para tablet PC: Windows XP sp3
Para PDA: Windows Mobile 5.0, 6.0 o 6.

2.3.3 Smart Device Framework v2.3

Smart Device Framework (SDF) permite implementar una serie de funcionalidades que no están presentes para la versión Compact Framework de Microsoft .NET (dotNet), pero si existen en la versión extendida de dotNet.

SDF proporciona un conjunto de librerías llamas OpenNetCF las cuales son extensiones del núcleo .NET Compact Framework Class Libraries. Y son utilizadas para proveer a dispositivos móviles, como PDAs o Smartphones, las mismas funcionalidades que a equipos que soportan la versión extendida del framework de dotNet.

Actualmente SDF está en su versión 2.3 y es compatible con dotNet Compact Framework 2.0.

2.3.4 SmartDraw

Programa creado por los alumnos del DCC (Universidad de Chile) Felipe Baytelman y Karl Strasser. SmartDraw permite realizar aplicaciones colaborativas, su característica principal es que las aplicaciones utilizan objetos compartibles en la red a través de XML y que todos los elementos son representados en pantalla a modo de elementos gráficos, además SmartDraw interpreta gestos realizados sobre la pantalla de los dispositivos como un evento realizado por el usuario.

3 Requerimientos del Sistema

Existe un conjunto de características mínimas que se deben cumplir para el desarrollo de una plataforma de esta índole.

3.1 Manejo de múltiples grupos

Los usuarios de hoy en día se han acostumbrado a atender a más de una tarea a la vez [29], considerando esta capacidad, la plataforma debe ofrecer soporte para que los usuarios se asocien entre sí, realizando grupos y creando jerarquías dentro de estas, permitiéndoles participar simultáneamente en más de un grupo.

3.2 Sincronización de objetos

Uno de los principales objetivos en una plataforma que este orientada a ambientes peer-to-peer es la transparencia de la comunicación y replicación de objetos [1].

Debido a que la plataforma proveer trabajo colaborativo en tiempo real es vital que la plataforma permita distribuir la información a través de la red, a la vez de mantener consistencia en la información entregada en cada nodo, todo esto con el fin de mantener el mismo estado, de los objetos compartidos, para todos los nodos de una sesión de forma transparente para el programador y el usuario.

3.3 Soporte de testeo

Es conocido lo complejo que es realizar la etapa de testeo en todo desarrollo de software, además de ser un punto crucial para ratificar la confiabilidad del software, es por esto y para facilitar esta etapa del desarrollo, que debe contar con herramientas para la evaluación de las aplicaciones y del comportamiento de la plataforma.

3.4 Almacenamiento de objetos

Se debe proveer administración de los objetos realizando una estructura de almacenamiento general.

Para desarrollar una plataforma colaborativa es necesario mantener todos los objetos compartidos en alguna estructura, ya que de esta forma es posible mantener la sesión, esta estructura y todas las herramientas anexas que faciliten el trabajo con los objetos compartidos son necesarias para tener una plataforma para aplicaciones colaborativas.

3.5 Interfaz usuaria por medio de gestos a mano alzada

Las actuales interfaces tipo escritorio de Windows no función bien para equipos con pantallas de pequeño tamaño [29], además para realizar una aplicación interactiva donde la dinámica del trabajo exige velocidad, los gestos permiten realizar en un solo trazo una serie de operaciones [22]. Es por ello que se debe realizar un mecanismo de interacción entre el usuario y el sistema utilizando gestos, de esta manera se maximiza el área de trabajo y se simplifica la interacción.

3.6 Permitir la escalabilidad del sistema

El desarrollo tendrá que considerar que la plataforma siempre estará en crecimiento, por tanto se debe entregar flexibilidad en el desarrollo, considerando que los objetos creados podrán ser extendidos (heredados), creado nuevas estructuras y reutilizando las existentes. Además se debe entregar modularidad y minimizar las dependencias, de tal forma que sea ordenada la comunicación entre capas.

3.7 Facilitar el mantenimiento del Software

Debido a que se debe tener presente que el área de la computación móvil está en constante cambio y es necesario adaptarse a las nuevas tecnologías que vayan apareciendo se debe considerar el mantenimiento a futuro del software.

Para lograr este requerimiento se debe cuidar la forma de programar, incluyendo descripción de cada método, realizando una documentación de la API, subdividiendo el sistema en capas, de forma que se pueda extender y testear cada capa sin la necesidad de tener las capas superiores.

3.8 Reducir los tiempos de desarrollo

La plataforma debe contar con un conjunto de herramientas que faciliten el desarrollo de aplicaciones, algunos elementos que se deben incluir son: documentación, herramientas para el testeo, aplicaciones de testeo, priorizar automatización de acciones del sistema. Todo esto con el fin de facilitar y apoyar la tarea de los programadores de aplicaciones.

3.9 Entregar un desempeño aceptable para los usuarios

La plataforma debe responder en tiempos inferiores a 2 segundos a las acciones que realizan los usuarios, esto tanto en red como localmente, una red será aceptable si cuenta con al menos 5 equipos funcionando, entre PDAs y TabletPCs, demostrando así que la plataforma supera a las pruebas realizadas en las plataformas estudiadas.

4 Diseño e Implementación

En esta sección se detallan como se llevó a cabo la plataforma, describiendo de manera técnica los módulos y subsistemas más relevantes de NetSketcher.

4.1 Estructura General de la plataforma

Basado en las estructuras de capas encontradas en las plataformas estudiadas se definieron 3 capas para NetSketcher:

- **Capa de Comunicación**, que permite el envió de paquetes a través de una red P2P entre dos nodos de la red o realizando multicast.

- **Capa de Administración de Sesión**, que administra sesiones colaborativas, controlando la interacción entre usuarios, realizando el almacenamiento de objetos y manteniendo la consistencia del estado de la sesión en general.

- Por último está la **Capa de Trabajo**, la cual entrega soporte a los programadores para desarrollar sus aplicaciones, administrando todas las funcionalidades de interfaz, realizando la captura y procesamiento de gestos desde pantalla y coordinando el envió y recepción de objetos sincronizados.

Figura 1: capas de la plataforma NetSketcher.

Debido a que la Capa de Comunicación es un proyecto externo a este, se procederá a explicar desde la capa de administración de sesión en adelante, considerando que en la sección 2 se explicó esta capa en detalle.

4.2 Diseño de la Capa de Sesión

Esta capa es la encargada de procesar la información que se recibe y envía a la red, además realiza evaluaciones de consistencia de los datos intercambiados y las políticas de acceso a los datos que tiene el usuario del equipo.

La capa fue dividida en 5 módulos:

- Sistema de administración de datos
- Sistema de sincronización de objetos
- Administración de sesiones de red y políticas de acceso
- Interfaz de acceso a la Capa de Comunicación
- Administración General de la capa de sesión

Además existe un conjunto de elementos que esta capa puede compartir en red, los cuales se proceden a definir a continuación.

4.2.1 Elementos colaborativos

Estos elementos que son la base del sistema colaborativo se dividen en dos tipos, los objetos compartibles (Sharable Objects), objetos que se mantienen sincronizados por la plataforma y los comandos remotos (Remote Commands), sistema de métodos que se pueden llamar y ejecutar de forma remota.

4.2.1.1 Los objetos compartibles o Sharable Objects

Los Sharable Objects están construidos basados unos en otros (ver Figura 2) otorgando las características de un tipo y agregando nuevas. Estos objetos compartibles son:

- **Sharable:** este tipo es la raíz de todos los objetos de la plataforma, está pensado para generar objetos que se desean compartir sin realizar ningún control de sesión, las propiedades más importantes de este objeto son, el ID y Creator, que respectivamente mantienen el identificador único de una instancia en la plataforma y el nombre del dispositivo donde fue creado.

  El método más importante de esta clase es PostProcess, este método es el que caracteriza a los objetos compartibles. Este método se ejecuta luego que se han realizado todos los procesos de evaluación de la capa de sesión y permite realizar ejecuciones remotas de un objeto de este tipo.

```
public virtual Sharable PostProcess(string idSender){…}
```

Código 2: método PostProcess de la clase Sharable.

- **Elemental:** el objetivo de Elemental es contener todas las propiedades que se necesitan para poder ser persistido, sincronizado y funcionar en una sesión especifica de red, pero sin la propiedad de ser gráfico, ya que algunos elementos pueden ser utilizados colaborativamente, pero no necesariamente deben ser presentados en pantalla, con esto se optimiza tanto el uso de memoria, como el consumo en procesamiento, al no almacenar propiedades gráficas inútiles y evitar evaluaciones propias de los elementos gráficos (ejemplo: al ser pintadas en pantalla o si deseamos verificar si tiene permisos de visualización).

  Las propiedades más importantes son, el atributo de versión, el id de la sesión a la que pertenece y el booleano que indica si es editable. Elemental y toda clase que se herede adquiere la propiedad de ser sincronizable y pertenecer a una grupo de la red.

- **Drawable:** tipo de objeto base para crear elementos gráficos en la plataforma, incluye todas las propiedades de los anteriores tipos y agrega algunos más, como visibilidad, indicador de selección y un marco que indica la posición, el ancho y alto en el cual se encuentra el objeto dibujable.

  En cuanto a métodos, incluye un conjunto de métodos necesarios para dibujar cualquier elemento en pantalla, algunos de estos son:

  - Draw, que permite dibujar lo que en él se implemente (ver código 2).
  - Touched, si el objeto ha sido interceptado por un rectángulo este método retorna un booleano (ver código 2).

```
public virtual void Draw(Graphics g, float offsetx, float offsety,
float zoom) { }

public virtual bool Touched(RectangleF r)
{
      if (!IsVisible())return false;
      return RectangleFTools.IntersectRects(_rect, r);
}
```

Código 2: métodos de la clase Drawable.

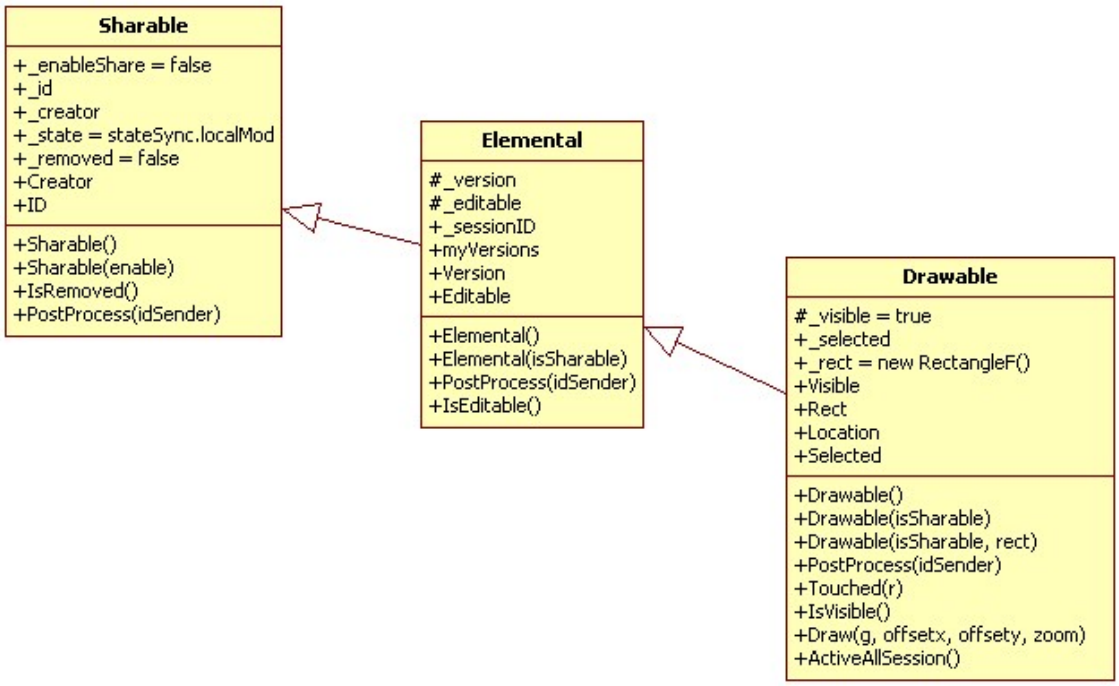


Figura 2: Diagrama de objetos de Share Objects.

4.2.1.2 Mensajes tipo Remote Commands

No hay duda que los objetos compartibles son claves en este sistema colaborativo, pero existen algunas cosas que no se pueden enviar por medio de estados, para ello existen los comandos remotos, llamados Remote Command, se caracterizan por que solo incluyen un método sencillo, llamado Execute.

```
public override object Execute(ArrayList list){…}
```

Código 3: métodos común de los objetos tipo RemoteCommand.

El método Execute recibe un listado al cual se pueden agregar objetos tipo Object. Luego al momento de ejecutarse en un dispositivo remoto, se llama al método permitiendo realizar todo tipo de acción implementada en él y cargar todo tipo de objeto en el listado, por último se detona un evento de arribo de un objeto tipo Remote Command, permitiendo realizar eventos de forma remota.

Por último señalar que dado que es una plataforma orientada a objetos gráficos, estos generalmente realizan una serie de cálculos cuando se rotan, se minimizan u otras transformaciones las cuales hacen perder fidelidad de su representación, es

por esto que todos los elementos que se crean utilizan float para dar la mejor precisión en los valores, debido a que la librería gráfica GDI+ de Microsoft que se encarga de los objetos gráficos como rectángulos, elipses o puntos no entrega soporte para float en su versión para compact framework, se creó un nuevo tipo de punto, llamado PointShr, se implementaron todos los métodos que existen para PointF (típico punto hecho con float de C#) y además se crearon todas las funcionalidades que existen para Rectangle para la clase RectangleF. Con estos elementos anexos se da soporte al uso de float.

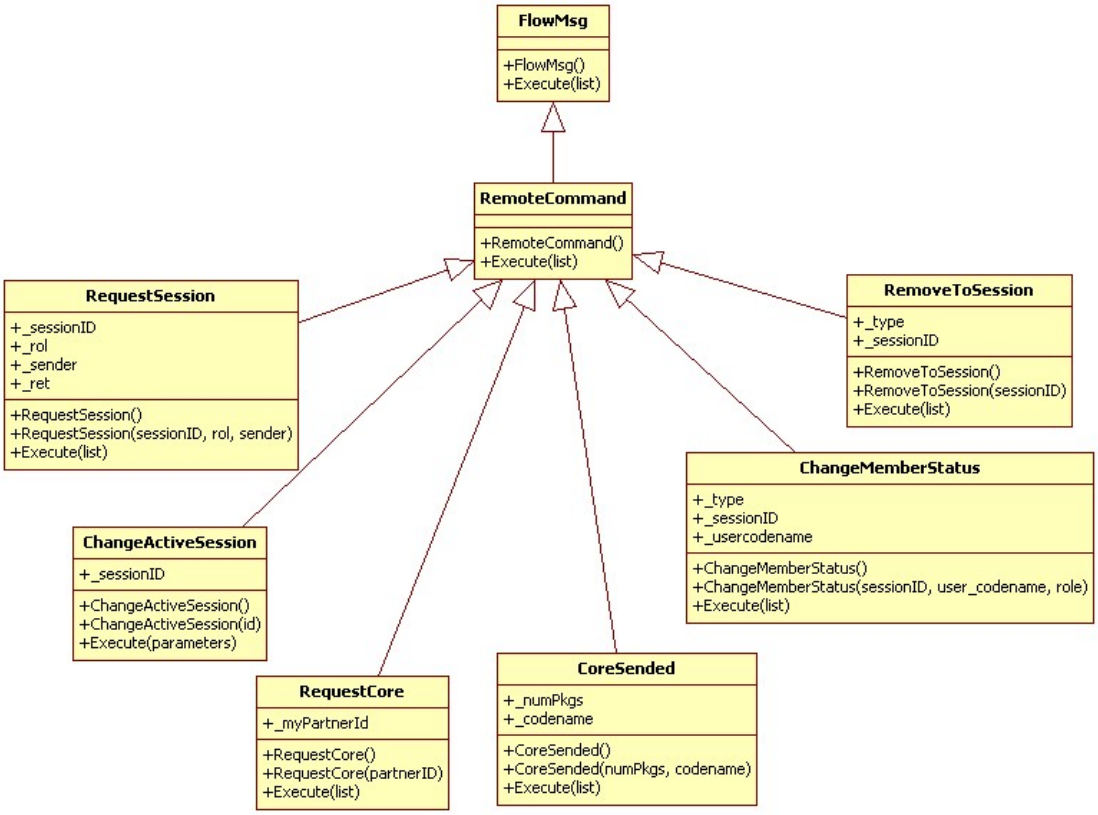


Figura 3: Diagrama de objetos de RemoteCommad y algunas de sus implementaciones.

4.2.2 Administración de Datos

Para realizar la administración de datos se desarrolló un repositorio dual, su principal característica es subdividir el espacio donde el usuario realiza las modificaciones del conjunto de objetos consistentes en red, esto agrega control en cómo y cuándo realizar modificaciones en red y a la vez permite realizar tareas en paralelo sobre los repositorios.

4.2.2.1 Funcionamiento del repositorio dual

El repositorio dual cuenta con dos estructuras tipo hash que permiten mantener el listado de objetos locales y compartidos. Cuando llega un objeto desde la red, este es procesado y almacenado en el repositorio de objetos consistentes, llamado Repository, a la vez es enviado al repositorio donde trabaja el usuario llamado Snapshot, esto se puede ver en la Figura 4 como las flechas que suben en el diagrama.

En el caso que un usuario cree o modifique un objeto, este no será sincronizado con el Repository hasta que el usuario o alguna lógica de capas superiores lo determine, permitiendo trabajar y realizar todas las modificaciones que se desee del objeto que *"vive"* en el Snapshot hasta el momento en que se sincroniza con el Repository y con toda la red (ver Figura 4), esto se logra gracias a que los objetos solo sincronizan su estado, así que no necesitan de los estados anteriores para construir la *"situación actual"* de un objeto, permitiendo restringir la actualización de un objeto tanto como determine el programador y dependiendo de las especificaciones que requiera su aplicación.

Otra característica del repositorio dual, es que los objetos son duplicados en los 2 repositorios, permitiendo realizar tareas en paralelo entre el Repository y el Snapshot, el primero permite realizar tareas de red, como establecer consistencia, verificar versión del estado de un objeto y establecer los permisos que el usuario tendrá sobre los objetos, el segundo repositorio permite realizar tareas de refresco de los objetos dibujables sobre el canvas y modificación por eventos causados por usuarios, permitiendo mantener una vista rápida de sus propias modificaciones.

El concepto de repositorio dual se basa en los cambios que un usuario realiza sobre un objeto, debido a que los usuarios suelen corrigen sus propias modificaciones y su primera decisión no es la última, no es necesario actualizar los cambios a la misma velocidad en la red, por esto el sistema dual permite hacer un filtro de cuando se actualiza en la red y bajo qué circunstancias, por supuesto esto es debido a la sincronización a través de estados, que evita requerir la suma de los cambios para construir la *"situación actual"* de un objeto, como lo haría la sincronización por eventos.

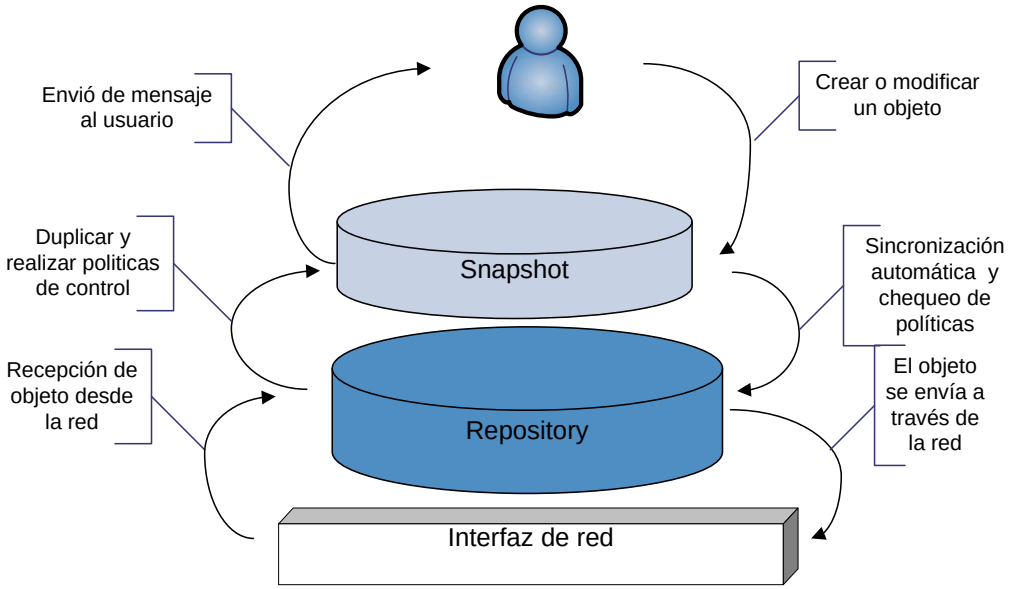


Figura 4: Diagrama que representa el sistema de repositorio dual.

4.2.2.2 Vistas

Además del repositorio dual, este módulo provee de un sistema de vistas del repositorio (View), el objetivo es extraer un sub conjunto de los objetos que existen en el repositorio, por lo general del Snapshot, el sistema de vistas permite trabajar con un conjunto de objetos a la vez, permitiendo a los programadores organizar la forma en que administran los objetos que *"viven"* en la sesión colaborativa de forma transparente.

Para obtener un conjunto de Share Objects, se debe crear una instancia de un View y colocar en el constructor una instancia de un objeto tipo Comparable (Ver Figura 4), los Comparables implementan la interfaz IComparable propia del lenguaje C#, esta contiene un único método que se encarga de realizar comparaciones para clasificar instancias de objetos, en particular para la plataforma objetos de tipo Sharables. Únicamente creando Views se puede trabajar con Share Objects.

```
View _users = new View(new AllParticipantComparable());
```

Código 4: ejemplo de instanciación de un objeto View
Usando un objeto Comparable se selecciona solo los objetos tipo Participant.

Utilizando Views se pueden agregar objetos a los repositorios, eliminar del repositorio y actualizarlos si se han hecho cambios, todo esto se hace directamente al repositorio Snapshot, pero se necesita de la ejecución del método de sincronización de repositorios que tiene Snapshot para sincronizar con el repositorio de consistencias (Ver código 7).

```
public void Add(Sharable shr){}

public static bool Update(Sharable shr){}

public bool Remove(Sharable shr){}
```

Código 5: métodos principales de View.

View es la forma predeterminada de almacenar y sincronizar los objetos en la red de manera transparente y sencilla.

4.2.2.3 Objetos Removidos

En el caso de los objetos que son eliminados de una sesión, estos son *"marcados"* para ser borrados para la siguiente sincronización, cuando la sincronización sucede, el objeto es enviado a un hash anexo al Repository, llamado TrashCan y eliminado de Repository y Snapshot, luego el objeto es enviado a los demás usuarios y se realiza el mismo procedimiento, eliminando el objeto de la sesión (Ver Figura 5).

Este sistema de *"aislar"* los objetos eliminados se utiliza para reducir el tamaño del listado de objetos sincronizados, con ello la cantidad de los objetos que son procesados o evaluados se reducen, descartando a los objetos que no son utilizados en la sesión. Sin embargo estos objetos eliminados pueden querer recuperarse en algún momento, es por esto que no desaparecen del todo y siempre es posible recuperar un objeto para volver a estar sincronizado.

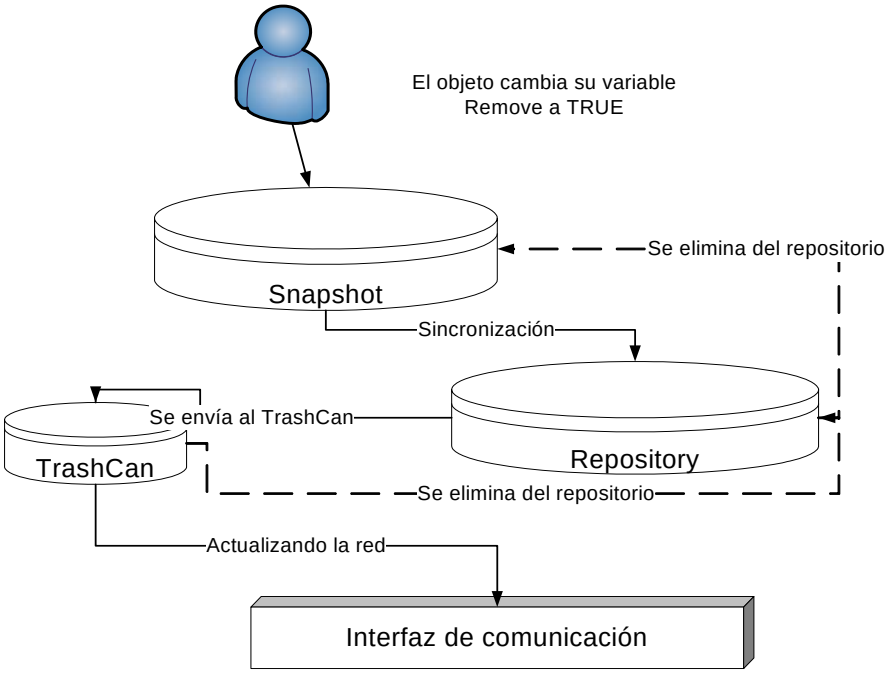


Figura 5: Representación del movimiento entre repositorios que realiza un objeto removido.

4.2.3 Preservación de consistencia de objetos en la red

Mantener el mismo estado para cada objeto en una red distribuida es complejo, sobre todo porque existen varios casos donde se producen conflicto en la información y es necesario decidir correctamente para mantener la consistencia.

En sistemas distribuidos el tiempo no es una medida absoluta para todos los dispositivos involucrados en la red, sin embargo podemos establecer que 2 eventos se realizan al mismo tiempo cuando consideramos a un objeto como una serie de eventos que lo van transformando, suponiendo que los eventos están numerados, entonces 2 eventos ocurren al mismo tiempo si estos tienen el mismo número identificador.

En el caso de la plataforma se estudiaron 2 casos donde suceden conflictos:

- Como los Share Objects son compartidos por estado, se pueden recibir instancias con estados de menor prioridad después de instancias de mayor prioridad, obteniendo estados finales distintos para usuarios distintos.
- El otro caso es cuando 2 usuarios modifican la misma propiedad de un objeto en el mismo momento produciendo 2 eventos con la misma prioridad.

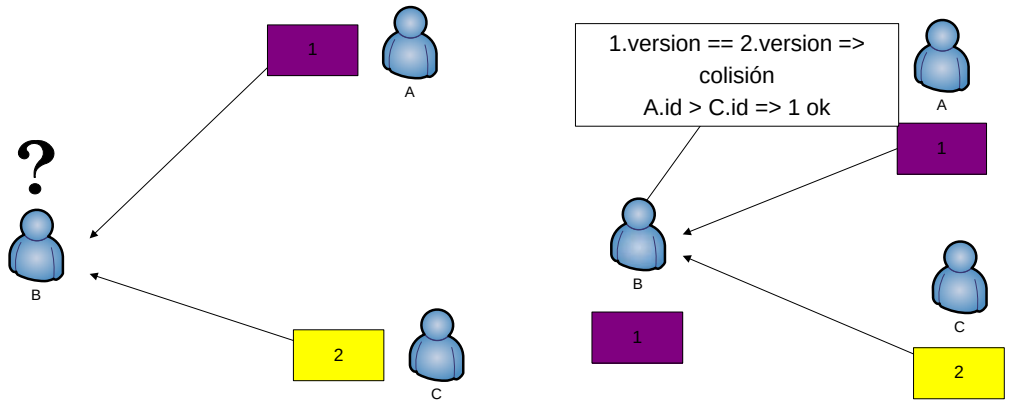
Para resolver estos casos se crea un sistema que decide si se modifica o no un objeto basado en la meta-información de este y que por supuesto acompaña al objeto a través de la red. La meta-información incluye un número de versión, el cual es evaluado cada vez que se recibe un objeto y comparado con el objeto almacenado en el repositorio de objetos consistentes (Repository), si es mayor que el estado que se tiene el objeto puede ser modificado, sino se desechará.

Cuando 2 usuarios realizan un evento sobre el mismo objeto al mismo tiempo, los objetos luego de chequear que el estado cumple con la condición de ser mayor que el del objeto almacenado, se procede a evaluar el listado de usuarios, los cuales tienen prioridad de modificación del objeto, luego el usuario con mayor prioridad modificará el objeto en todos los equipos y los equipos en los cuales el usuario con menor prioridad entregó el mensaje antes verán como el objeto pasa por ese estado y luego cambia al estado entregado por el usuario de mayor prioridad, de igual forma el usuario de menor prioridad presenciará la misma situación. Este sistema de resolución de prioridad asegura la rotación de prioridades para no entregar mayor control a un usuario por sobre otros y utiliza un sencillo indicador para confirmar que el listado de usuarios en cada equipo es el mismo.

Un ejemplo sencillo para entender esto es suponer que existen 3 usuarios (A,B y C) conectados en la red y los 3 ven un rectángulo que esta dibujado en pantalla, el usuario A cambian el color del rectángulo a morado y el usuario C lo cambia a amarillo (Ver Figura 6.a).

Luego es necesario que para los 3 usuarios tengan el mismo color del rectángulo para que sea considerado sincronizado. Entonces para el usuario B el sistema revisa la versión de los objetos, los cuales tienen la misma versión identificándose una colisión, luego para decidir, el sistema revisa los usuarios, el que tenga mayor prioridad podrá modificar el color del objeto, en la Figura 6.b se puede ver que el usuario A tenia mayor prioridad, cambiando el color a morado.

Para el usuario C de la Figura 6, el sistema deberá resolver el mismo problema obteniendo un cambio de amarillo a morado.



(a) (b)

Figura 6: Ejemplo de solución de inconsistencia, (a) dos usuarios modifican la misma variable al mismo tiempo, (b) para un tercer usuario se decide la colisión por el usuario de mayor prioridad.

Cabe destacar que el sistema de sincronización utiliza un indicador de que variable o clase se debe mantener un control de versión, para ello basta con agregar una etiqueta sobre la variable o clase, para que el sistema automáticamente identifique este elemento para ser controlado.

```
[VersioningAttribute()]
public RectangleF _rect = new RectangleF();
```

Código 6: ejemplo de cómo utilizar la etiqueta de versionamiento.

4.2.4 Grupos y permisos de acceso

Este módulo está encargado de gestionar los roles y los grupos que se puedan crear además de establecer los permisos que tienen los usuarios basados en el rol al cual pertenezcan.

El sistema de grupos se compone de 3 elementos Session, Role, Right. Cada grupo cuenta con una instancia de Session, una instancia de Role por cada rol que se defina y un Right que contiene las políticas de accesos para todos los roles.

Los grupos incluyen un único administrador el cual se encarga de aceptar, invitar o rechazar a usuarios que desean participar de la sesión.

Los Right realizan la lógica cuando se desea sincronizar un objeto en la red, ya sea cuando se hace la sincronización de repositorios o cuando arriba un objeto desde la red.

Los Right indican las políticas de acceso que tendrá el usuario del dispositivo sobre el objeto. Las políticas de acceso se basan en las combinaciones posibles de dos características de los objetos, visibilidad y edición, estas ofrecen 4 posibles políticas:

- Visible pero no editable
- Visible y editable
- Invisible y editable
- Invisible y no editable

Los tipos de roles se establecen basados en las políticas de acceso y cada uno utiliza una sola política:

- Administrador
- Participante
- Invitado
- Inhabilitado

Sin embargo, establecer los permisos que tiene cada rol depende de que se indique en un Right, por tanto los permisos que estos roles dependen de cómo se definan en el Right.

Para los usuarios tener permisos de accesos a los objetos depende de a que rol pertenezcan, estos solo pueden pertenecer a un único rol por grupo, pero pueden cambiar de rol dentro del grupo.

En el caso que un usuario este en más de un grupo, el usuario podrá tener roles distintos en distintos grupos.

Para acceder a un grupo se realiza por medio de peticiones y el administrador es quien decide agregar o no a un usuario, además de acceder a un grupo se puede pedir cambiar de rol o simplemente renunciar al grupo. Todas estas acciones pasan por el administrador quien da los permisos.

A diferencia del sistema de manejo de objetos, este sistema no es de pares, ya que todo el control del grupo pasa a través de un solo usuario, el administrador, es él quien toma todas las decisiones y para realizar esto se utiliza solo comandos remotos.

4.2.5 Interfaz de acceso a la capa de comunicación

La interacción con la capa de comunicación se aisló en un módulo independiente, así se simplifica la actualización y reduce los conflictos que puedan suceder al cambiar la capa de comunicación.

Este módulo cuenta con una clase encargada de inicializar la comunicación y la configuración de la red, además incluye los métodos de envió de datos por la red. También existen 3 interfaces que permiten implementar métodos que capturen los eventos que se emiten desde la capa de comunicación.

La comunicación cuenta con 2 protocolos utilizados en la capa de comunicación, según sea el tipo de mensaje que se desea envía:

- **Flujo de datos:** protocolo encargado de transmitir Remote Commands y mensajes de bajo nivel del sistema, estos no son persistentes en la sesión y solo envían una orden de ejecución de un método de forma remota.
- **Sincronización de objetos:** protocolo encargado de transmitir objetos que se sincronizan y persisten en la sesión colaborativa, además este protocolo admite enviar más de un objeto por mensajes.

Los protocolos están implementados como un conjunto de tipos de mensajes que pueden ser generados según determine el protocolo utilizado. A través de estos tipos de mensajes los objetos son serializados y deserializados, utilizando serialización XML.

Este módulo de comunicación es el encargado de evaluar si un objeto es compartible en red o es local, esto se realiza filtrando los objetos locales al momento de enviar un mensaje a través de la red. Además los métodos de comunicación para sincronización de objetos agrupan un lista de objetos en un mismo mensaje hasta un máximo de 20 objetos, esta estrategias apoya la homogenización del tamaño de los mensajes que se envían a través de la red.

4.2.6 Administración general de la capa de sesión

La capa de sesión cuenta con un administrador general de la capa llamado Environment Session Layer, es la interfaz que permite la interacción con capas superiores, entregando un conjunto de eventos para cada acción que sucede en la capa, métodos que permiten el acceso a la información y métodos para transmisión de información.

Dentro de este administrador están incorporados todos los procesos realizados por los otros módulos, realizando evaluaciones de consistencia, sincronizando repositorios y chequeando las políticas de acceso.

Las principales funciones que se pueden realizar a través de este administrador son la sincronización de repositorios y el envió de mensajes remotos (Remote Commands).

```
public bool SyncRepository(){…}

public void SendCommand(RemoteCommand com){…}
```

Código 7: métodos que permiten acceso a la red

En resumen este administrador se encarga del acceso a la información y permite la manipulación de esta para cualquier aplicación que utilice esta capa.

4.3 Diseño de la capa de Espacio de trabajo

Esta capa está encargada principalmente de la captura de eventos realizados por el usuario y la presentación en pantalla de las aplicaciones. También está encargada de la administración de eventos y la entrega de estos a las distintas aplicaciones que estén activas, cuenta con un conjunto básico de objetos y un listado de gestos.

4.3.1 Estructuras primarias para crear una aplicación

Esta capa cuenta con 4 estructuras imprescindibles y que suelen interactuar estrechamente entre sí.

- **Sketches:** objetos gráficos que heredan de Drawables, permiten crear elementos gráficos que son compartibles en la red, es la base de la que se heredan todos los objetos gráficos de esta capa.
- **Modes:** Encargados del manejo general de todos los eventos que puede capturar una aplicación, este tipo de objeto entrega la estructura base para realizar una aplicación.
- **Gestures:** son objetos encargados del reconocimiento de un gesto realizado por el usuario y luego lo interpreta como una acción relativa a la aplicación que está activa.

- **SketchesContainer:** herramienta que encapsula todas las funcionalidades de una View, combinado con un PageSketch (ver la sección 4.3.2) crean una estructura análoga a un sistema de directorios o carpetas.

Estos 4 elementos interactúan entre sí frecuentemente y conforman la base de toda aplicación.

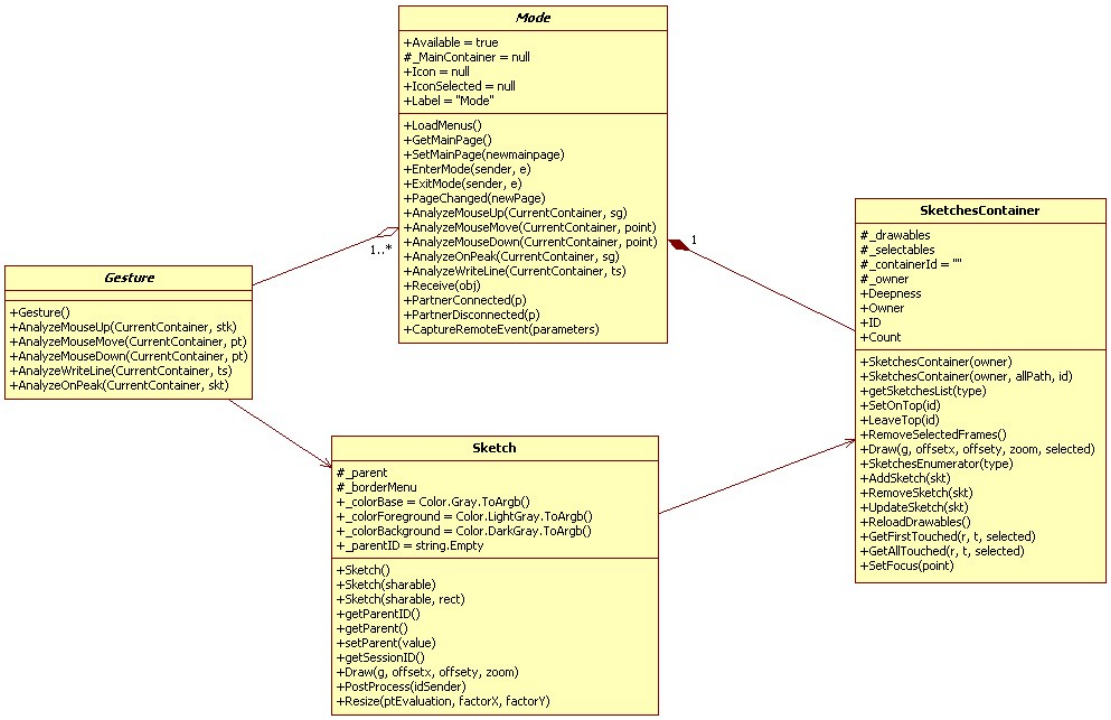


Figura 7: Diagrama de objetos que representa la relación entre Mode, SketchesContainer, Gesture y Sketch.

Cada gesto realizado por un usuario genera una acción que es capturada por el Mode activo y luego procesada por un Gesture, la lógica de este permite crea o modifica Sketches, luego para ser persistidos los cambios se almacena en un SketchesContainer el cual por lo general reside en el Mode activo. Por último si los Sketch son de tipo compartible en red, son sincronizados en la sesión.

4.3.2 Objetos básicos

La plataforma ofrece 4 objetos de tipo Sketch:

- **PageSketch:** Este Sketch tiene la capacidad de anidar Sketch, permite generar espacios de dibujo (Pages) anidados, los cuales permiten ampliar la cantidad de información que contiene un Sketch, los PageSketch cuentan con

un listado de estructuras llamadas SketchesContainer los cuales se encargan de administrar Views.

- **ImageSketch:** Como su nombre lo dice, es la estructura que se encarga de representar imágenes en la plataforma
- **TextSketch:** Encargado de representar texto como un objeto de la plataforma.
- **Stroke:** es la representación de trazos a mano alzada en la plataforma

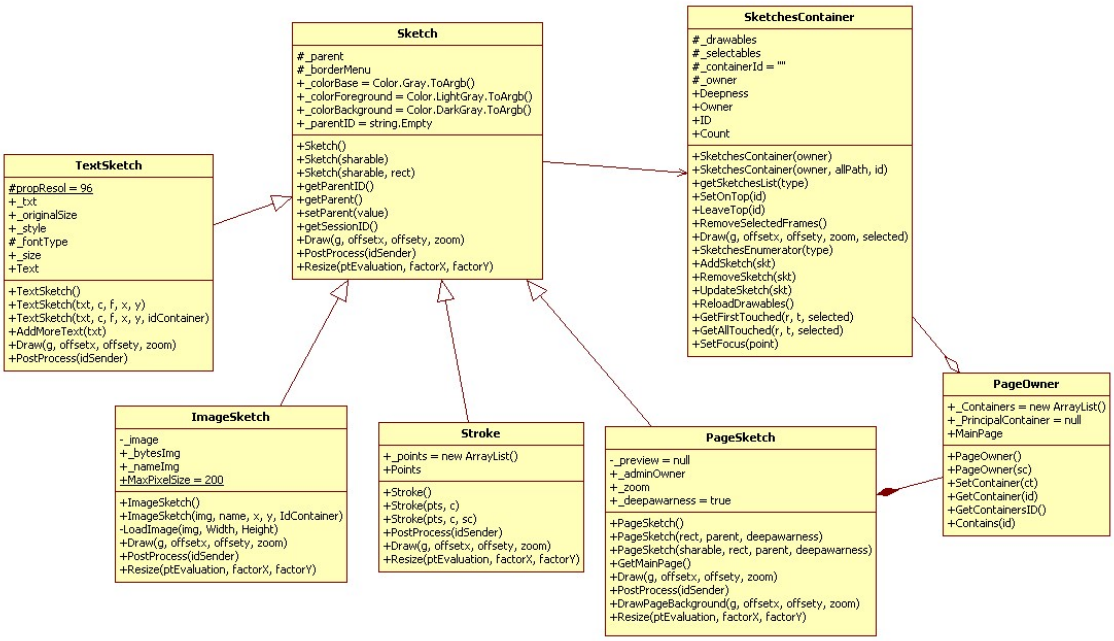


Figura 8: Diagrama de objetos que heredan de Sketch

Con estos Sketches se pueden construir todos los nuevos Sketch que se deseen heredando sus propiedades.

4.3.3 Gestos funcionales y de reconocimiento

Los gestos o Gestures son la forma principal de interacción que tiene el usuario con las aplicaciones, estos son todos construidos con la misma estructura, sin embargo se pueden dividir en 2 tipos:

- Los de reconocimientos, que se encargan de procesar los movimientos que realizó el usuario sobre la pantalla y decidir si cumple o no con la condición de ser un gesto.

- Los gestos funcionales, utilizan los gestos de reconocimiento y se encargan de realizar alguna acción sobre la plataforma, este tipo de gestos realiza un procedimiento siempre que el gesto de reconocimiento es aceptado.

Los Gestures se basan en una estructura genérica, son instanciados automáticamente por referencia y el programador solo necesita encargarse de implementar los métodos que necesita e indicar en el Mode donde va a utilizar este Gesture.

Existen 5 métodos generales para todo Gesture estos son:

AnalyzeMouseDown: captura la información realizada por el usuario luego que este presiona el mouse o el stylus sobre el canvas.

```
public virtual bool AnalyzeMouseDown(SketchesContainer CurrentContainer,
PointShr pt) {…}
```

Código 8: firma del método AnalyzeMouseDown

AnalyzeMouseMove: captura la información cuando se está realizando movimientos de mouse o stylus.

```
public virtual bool AnalyzeMouseMove(SketchesContainer CurrentContainer,
PointShr pt) {…}
```

Código 9: firma del método AnalyzeMouseMove

AnalyzeMouseUp: el cual captura la información luego que el mouse o stylus es soltado o levantado.

```
public virtual bool AnalyzeMouseUp(SketchesContainer CurrentContainer,
Stroke stk) {…}
```

Código 10: firma del método AnalyzeMouseUp

AnalyzeOnPeak: captura la información mientras se está moviendo el mouse o stylus, este sistema de captura, utiliza un punto máximo de movimientos realizados para gatillar el evento, indicado en el Mode activo.

```
public virtual bool AnalyzeOnPeak(SketchesContainer CurrentContainer,
Stroke skt) {…}
```

Código 11: firma del método AnalyzeOnPeak

AnalyzeWriteLine: captura la información cuando se escribe con teclado, la información se captura hasta el momento que se presione la tecla Enter.

```
public virtual bool AnalyzeWriteLine(SketchesContainer CurrentContainer,
TextSketch ts) {…}
```

Código 12: firma del método AnalyzeWriteLine

La plataforma cuenta con un conjunto de gestos de reconocimiento y funcionales, los cuales se detallan a continuación:

- **CirculeGesture:** reconoce el movimiento que realice el usuario sobre pantalla en forma de círculo, además determina si este se realiza en una u otra dirección.

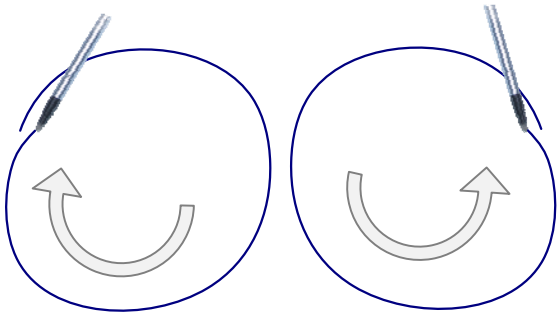


Figura 9: Ejemplo de cómo se debe trazar un CirculeGesture, el sistema reconoce si el gesto fue hecho en la dirección mostrada en (a) o en (b).

- **CrossGesture:** gestos que reconoce el movimiento de cruz sin soltar el lápiz.

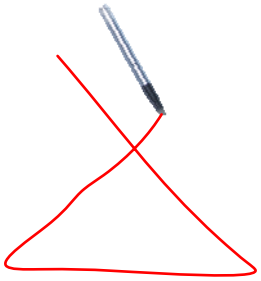


Figura 10: Ejemplo de cómo se debe trazar un CrossGesture.

- **DoubleClickGesture:** reconoce el gesto de doble clic, determinado por un cierto intervalo entre el primer y el segundo clic.

- **DoubleSurroundGesture:** reconoce el intento de encerrar un objeto con un movimiento doble o superior, del stylus o mouse, en torno al objeto a encerrar.

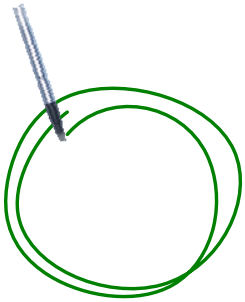
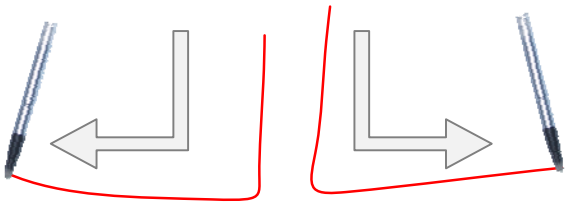


Figura 11: Ejemplo de cómo se debe trazar un DoubleSurroundGesture.

- **RectGesture:** reconoce la creación de un ángulo recto, este puede realizarse desde arriba abajo y continuar de izquierda a derecha (Figura 12.b) o continuar de derecha a izquierda (Figura 12.a), el gesto identifica uno u otro movimiento.



(a) (b)

Figura 12: Ejemplo de cómo se realiza RectGesture, el sistema reconoce si la línea horizontal es de derecha a izquierda (a) o de izquierda a derecha (b).

- **ResizeGesture:** reconoce la acción de reducir o agrandar el marco de un objeto gráfico y deriva la lógica al o los objetos afectados quienes determinar cómo realizar sus respectivos cálculos de tamaño.

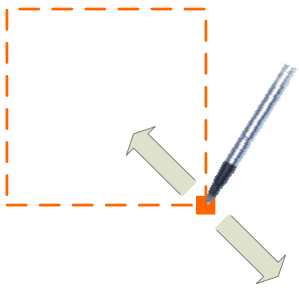


Figura 13: Ejemplo de cómo se realiza ResizeGesture, permite mover hacia adentro o hacia afuera el recuadro inferior derecho cambiando el tamaño del objeto seleccionado.

- **RightButtonGesture:** reconoce la acción de mantener presionado por un período de tiempo el mouse o stylus, similar al gesto de botón secundario utilizado en equipos con touchscreen.
- **TriangleGesture:** reconoce la acción de formar un triángulo equilátero, este puede tener su base de forma vertical u horizontal y puede estar dispuesto según se ve en las Figuras 14.a, 14.b, 14.c y 14.d, cada una de estas disposiciones del triángulo puede ser identificado por el gesto TriangleGesture.

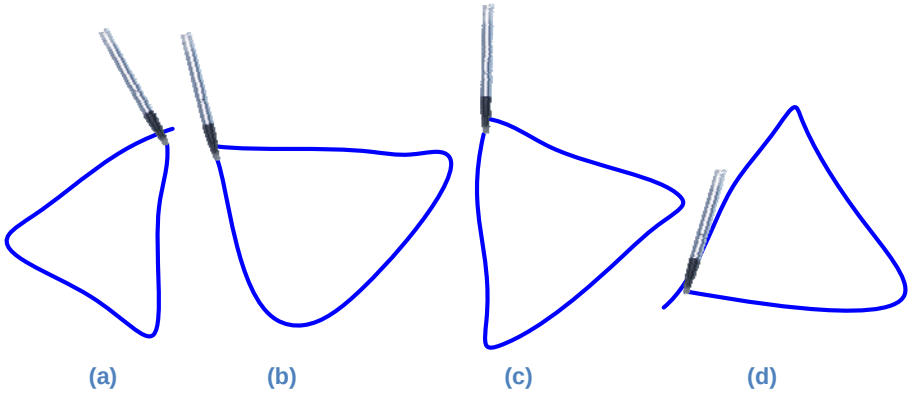


Figura 14: Los 4 tipos de TriangleGesture que se pueden realizar, (a) base vertical derecha, (b) base horizontal superior, (c) base vertical izquierda, (d) base horizontal inferior.

- **YesGesture:** reconoce la acción de check o ticket sobre la pantalla como se ve en la Figura 15.

Figura 15: Ejemplo de cómo se debe realizar un YesGesture.

Los siguientes gestos son funcionales y se generar a partir de los anteriores:

- **CreateRectNodeGesture:** Gesto que utiliza RectGesture para crear un objeto de tipo PageSketch.

- **CutGesture:** Gesto que utiliza CrossGesture para borrar todos los objetos en el canvas que intercepten con la cruz realizada.

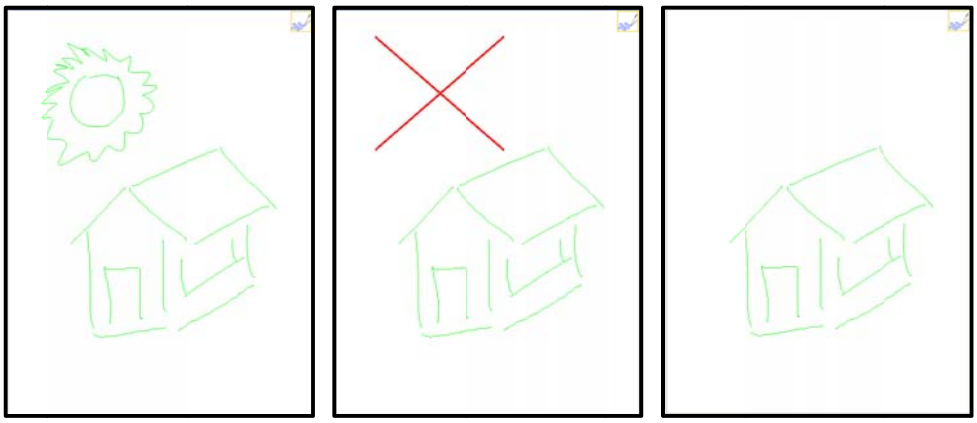


Figura 16: Ejemplo de cómo utilizando el gesto CutGesture se elimina un objeto en la pantalla, (a) estado previo, (b) el gesto es reconocido, (c) el objeto ha sido eliminado de la pantalla.

- **DragDrawableGesture:** Gesto que permite mover objetos en el canvas
- **EnterExitGesture:** Gesto que utiliza el DoubleClickGesture para entrar o salir de un Page, se realiza haciendo doble clic sobre un PageSketch para entrar y un doble clic en el canvas para salir.

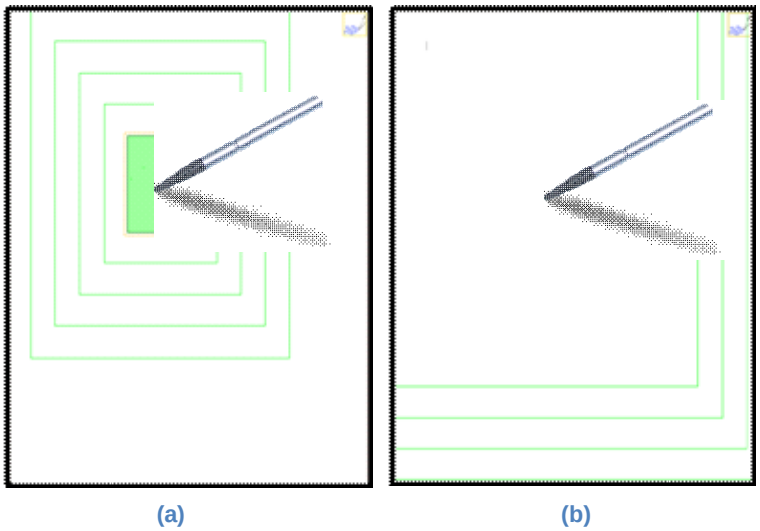


Figura 17: Ejemplo de cómo se utiliza el gesto EnterExitGesture para entrar y salir de un PageSketch. (a) Doble clic sobre un PageSketch entra al Page, (b) doble clic sobre el espacio vacío vuelve al Page anterior.

- **SelectionGesture:** utilizando un clic como gesto, selecciona o deselecciona un objeto al realizar un clic sobre este.

- **SetFocusPanningGesture:** permite realizar scrolling sobre el canvas cuando este es superior al viewport que se tiene del canvas.

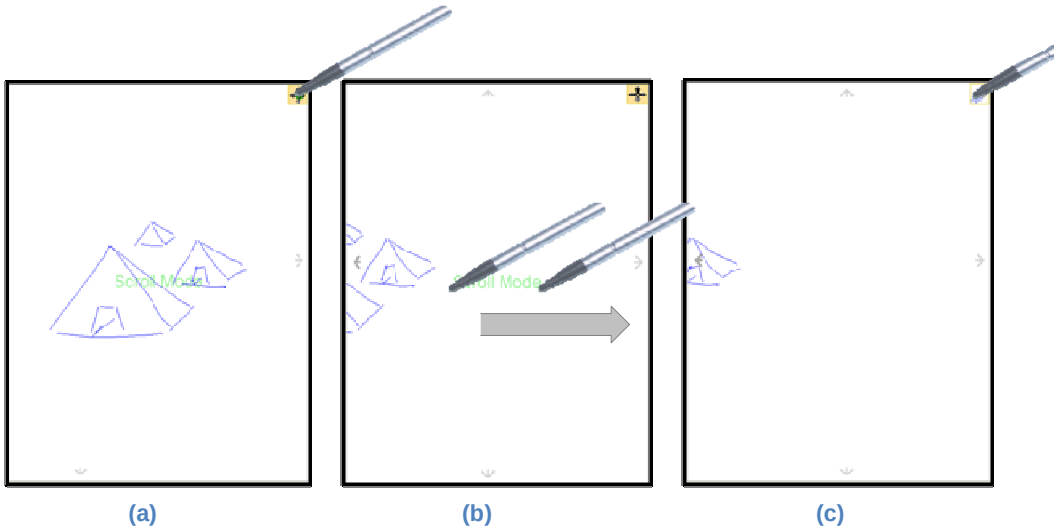
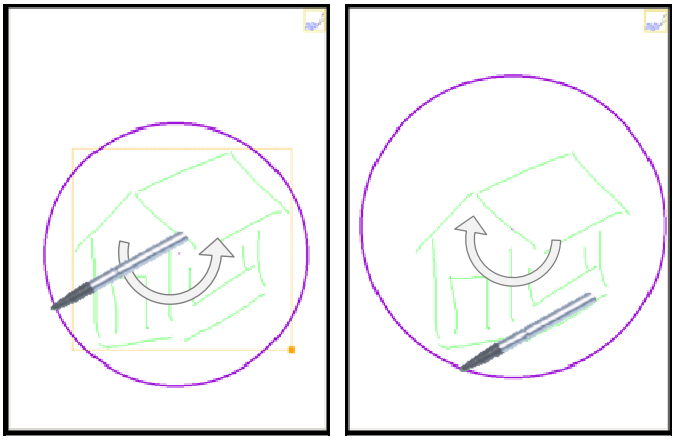


Figura 18: Ejemplo de cómo se utiliza el gesto SetPanningGesture, (a) antes de utilizar se realiza RightClickGesture sobre el ícono del menú,(b) arrastrando el stylus se desplaza dentro del Page,(c) para volver a editar se utiliza RightClickGesture otra vez.

- **StrokeGesture:** este gesto crea trazos como objetos, llamados Strokes, por lo general se agrega como el último gesto a ser evaluado en un Mode, ya que no tiene ninguna precondición para realizar su acción.

- **SurroundGesture:** utiliza el CirculeGesture para encerrar un grupo de objetos y seleccionarlos al realizar un círculo en el sentido opuesto a las manecillas del reloj y deseleccionarlos cuando el movimiento es en dirección de las manecillas del reloj.



(a) (b)

Figura 19: Gesto SurroundGesture , en (a) se realiza selección y en (b) deselección.

Un método que es utilizado recurrentemente en los gestos es Compress, método que reduce el conjunto de puntos capturados de la acción de un usuario, este sistema filtra los puntos que forman ángulos que sean mayores que el ángulo MaxAngle (ver Código 13) y menores que el ángulo MinAngle (ver Código 13). Para formar cada ángulo se utilizan 3 puntos consecutivos del trazo capturado y con estos se realiza la siguiente operación:

$$Dif_{angulos} = \left| arctg\frac{y_1 - y_0}{x_1 - x_0} - arctg\frac{y_2 - y_1}{x_2 - x_1} \right|$$

$$angulo = Min(2 * \pi - Dif_{angulos}, Dif_{angulos})$$

Basados en los ángulos encontrados en el conjunto de puntos se *"comprime"* la cantidad de puntos que representan el trazo lo cual suaviza las curvas y simplifica el análisis.

```
public static ArrayList Compress(ArrayList points, double
MaxAngle, double MinAngle){…}
```

Código 13: firma del método Compress.

4.3.4 Los modos de una aplicación

Los Modes o modos son objetos creados por referencia en la plataforma, son los contenedores que aglutinan todos los componentes que se utilizan en una aplicación, los Modes contienen un listado de interfaces que reciben distintos eventos, también

cuentan con atributos para crear un acceso directo en el menú principal y permite indicar que gestos se van a utilizar en la aplicación. Todo Mode cuenta con un ícono en el menú principal y un contenedor de Sketch que actúa como contenedor principal del modo.

En una aplicación usual debe existir al menos un Mode, pero puede tener tantos Modes como se desee, ya que los datos pueden seguir siendo trabajados por el usuario en otros modos.

Un Mode está pensado para ser implementado de forma sencilla, casi todas sus funcionalidades operan de una manera predefinida, por ejemplo la evaluación de los diferentes gestos del modo se realizan automáticamente, aun así el programador, en la medida que desee tener mayor control, puede sobrescribir las funcionalidades automatizadas.

Para crear un Mode sencillo se requiere extender de la clase Mode, hacer un constructor que no se le entreguen variables y dentro de este escribir el nombre que se desea colocar al Mode, un ícono que lo represente y los gestos que se desean utilizar (Ver Código 14).

```
public class MyMode : Mode
    {
        public MyMode()
        {
            Label = "My Mode";
            Icon = Resources.sketching3;
            _MyGestures.Add(typeof(CutGesture).Name);
            _MyGestures.Add(typeof(StrokeGesture).Name);
        }
    }
```

Código 14: ejemplo básico de implementación de un Mode, en este ejemplo se ha creado un Mode, llamado My Mode, que permite crear trazos (Stroke) y borrarlos haciendo el gesto de cruz.

Un Mode cuenta con muchos métodos que pueden ser sobrescritos, existen métodos que permiten realizar una acción cuando existe un nuevo usuario en la sesión o cuando es removido de la sesión, cuando se recibe un objeto o cuando se recibe un evento, para cargar menús extras, cuando se cambia de página, cuando está inactivo, cuando se carga una sesión desde disco o cuando se guarda la sesión a disco. La mayoría de las acciones que suceden en la plataforma pasan por el Mode, así el programador de la aplicación puede implementar lo que necesite sobre esta estructura.

4.3.5 Otras estructuras de la plataforma

NetSketcher cuenta con un conjunto de elementos que apoyan la usabilidad de las aplicaciones, entre eso tenemos los objetos de menú, las alertas y tips.

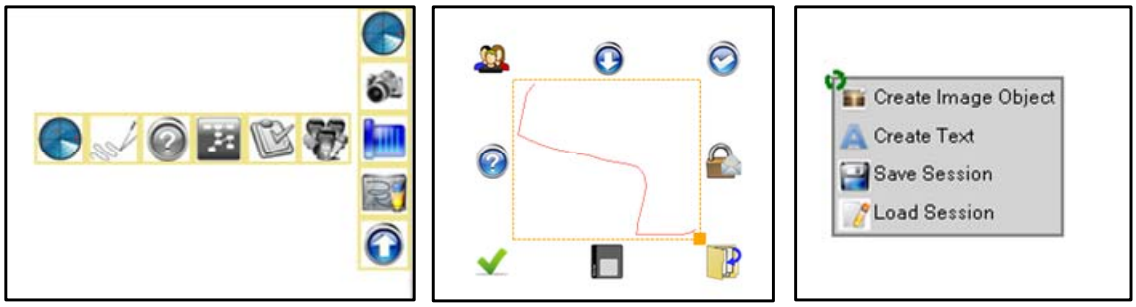
Las alertas y tips son formas de entregar información a los usuarios mostrando un mensaje sencillo en pantalla, este tipo de herramientas se utilizan para generar awareness sobre los sucesos que ocurren en una sesión, por defecto la plataforma utiliza alertas para indicar los cambio de estado de la comunicación.

4.3.5.1 Menús

Para apoyar la usabilidad de las aplicaciones existe un grupo de elementos diseñados para implementar menús, todos los objetos de este tipo son locales y se heredan de Drawable, el objeto padre de todos se llama LGraphicComponent, de este se extiende la clase LMenu y LButton de los cuales incluyen una serie de propiedades comunes para estos 2 tipos de estructuras.

Existen 3 tipos de menús, los de borde, de barra y de contexto.

- **Menús de Borde (LMenuBorder):** todo Sketch tiene un conjunto de botones que aparecen en torno a los Sketch cuando son seleccionados. Estos botones se incluyen en un LMenuBorder, este tipo de menú es cargado en cada Sketch y puede ser personalizado por el programador para cada nuevo tipo de Sketch.
- **Menús de Contexto (LMenuContext):** este tipo de menú aparece cuando el usuario mantiene pulsado el stylus y se gatilla el gesto de RightButtonGesture, LMenuContext es un menú inspirado en el menú contextual que se utiliza en S.O., este menú entrega funcionalidades generales enfocadas al espacio de trabajo que está activo, algunas de estas funciones son, guardar estado de la sesión y agregar un texto al Page actual.
- **Menús de Barra (LMenuBar):** por defecto la plataforma utiliza un menú de este tipo como menú principal, este menú principal entrega funcionalidades básicas que proveen usabilidad a las aplicaciones y contiene un botón por cada Mode de la aplicación. Además en cada Mode está integrado un método que permite cargar menús de barra. Por último se debe mencionar que este tipo de menús tiene la propiedad de carga menús anidados.



(a) (b) (c)

Figura 20: Tipos de menús que se utilizan en la plataforma, (a) menú de barra,(b) menú de borde,(c) menú de contexto.

Cada tipo de menú tiene un tipo especializado de botones, pero todos heredan desde LButton, aun que existen estos botones para cada menú, los menús aceptan cualquier LGraphicComponent como componente, esto con el objetivo de ofrecer compatibilidad con cualquier otro elemento que se implemente tipo LGraphicComponent y que se desee cargar en un menú.

Todos los LGraphicComponent descritos anteriormente cuentan con 4 tipos de eventos: clic, doble clic, un clic de largo periodo (RightClick) y enlazar a un Sketch (Link), estos eventos son procesados por una estructura contenedora de todos los objetos LGraphicComponent que es gestionada por el administrador de la capa.

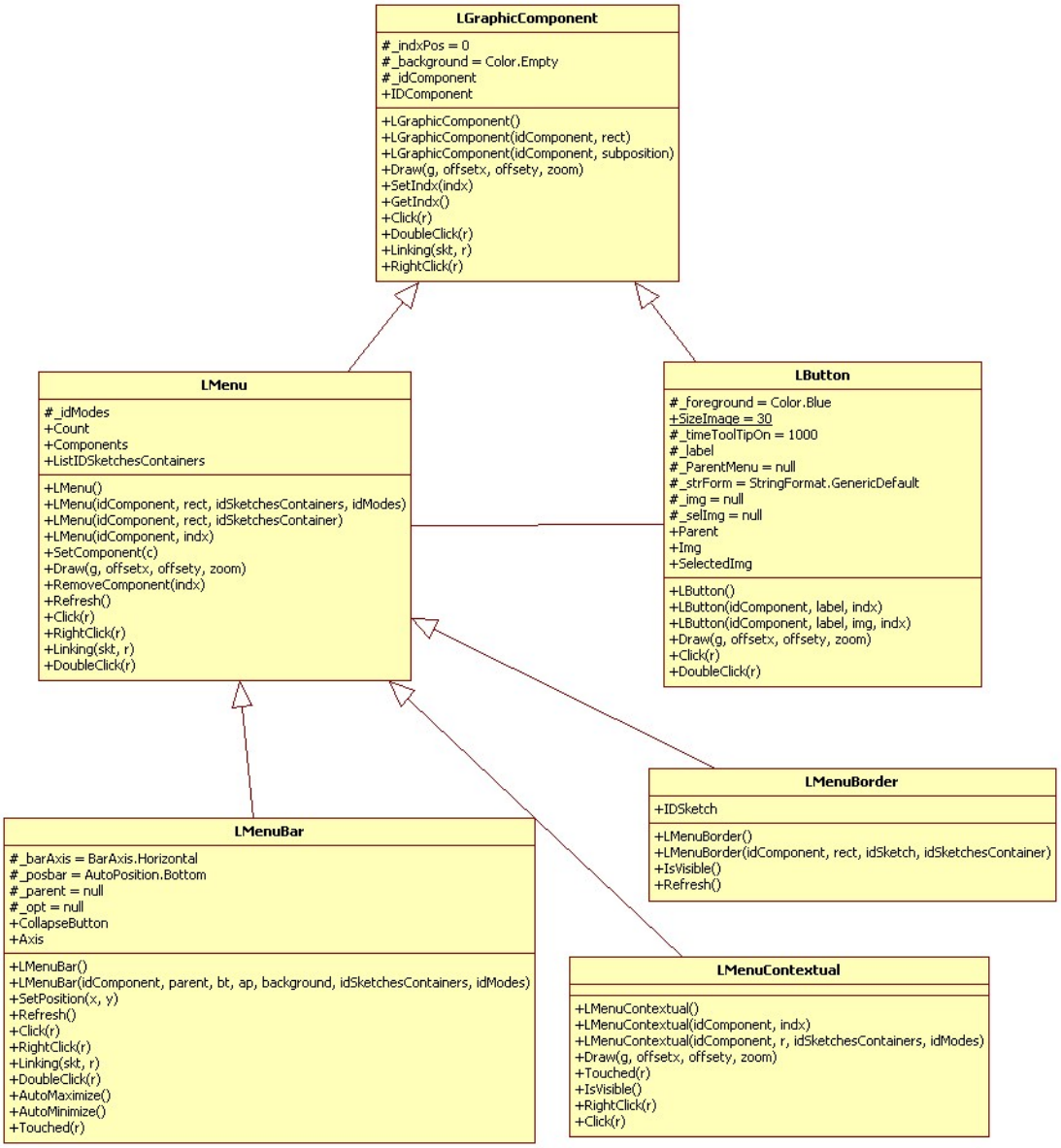


Figura 21: Diagrama de clases con la organización de los objetos LGraphicComponent.

4.3.6 NetSketcher: administrador de capa

La clase NetSketcher es el núcleo de la capa, encargada de la inicialización de todos los elementos de la plataforma, la captura de eventos y todo el procesamiento de pintado de la interfaz gráfica, cuenta con todas las herramientas para llamar a las instancias de cada objeto creado en el sistema, como Modes, Gestures, menús, etc.

Esta clase es la base de la capa y es la encargada de indicar que aplicación está activa y redirigir todos los eventos al Mode activo de la aplicación. También se encarga de pintar todo en pantalla y le ofrece a los programadores métodos que le permiten agregar cambios en la pantalla y priorizar que pintar.

```
public static void Paint(Type staticClass, string method, Object[]
parameters)

public static void Paint(Type staticClass, string method, Object[]
parameters, int duration)
```

Código 15: métodos para pintar en pantalla, requieren la clase, método y las variables del método que se llamará por referencia, el 1er método permite pintar instantáneamente y el 2do durante un periodo de tiempo.

NetSketcher también es el encargado de la sincronización de repositorios, llamado sincronización diferenciada, esta estrategia de sincronización utiliza dos condiciones, un Timer que determina que cada 0.2 segundos después de cada evento de mouse se realiza una sincronización, este periodo ofrece una *"ventana"* de tiempo en el que el usuario puede realizar modificaciones y de esta forma se anula la sincronización por otros 200 milisegundos, la otra condición es cuando se hace un máximo de eventos de mouse, cada 200 acciones de mouse se sincroniza automáticamente, este sistema evita el bloqueo de envíos por parte del sistema anterior.

5 Aplicaciones y Complementos

En esta sección se explican todos lo que se ha implementado utilizando la plataforma, además de todas las herramientas de apoyo que se incluyeron.

5.1 Aplicaciones Implementadas

5.1.1 FlowChart Maker

FlowChart Maker es una aplicación para gestión de procesos de negocio (Business Process Management o BPM) pero utilizando las cualidades de la computación colaborativa, esta aplicación apoya la dinámica de una reunión de planeamiento de un proyecto. Por su simpleza esta aplicación está enfocada para las instancias en las que se realiza el levantamiento de un proyecto.

FlowChart Maker crea diagramas de flujo con la notación que se utiliza en BPM, usualmente estos diagramas se realizan en un PC por una sola persona, pero utilizando la plataforma, la aplicación FlowChart Maker realiza esto de forma colaborativa. Este sistema permite realizar las correcciones en el momento y confluyen el aporte de conocimientos y experiencia de distintas personas, lo que ofrece un resultado más preciso para continuar el proyecto.

Otra característica que diferencia a FlowChart Maker de otras aplicaciones de BPM, es la forma de presentar los roles involucrados en el proceso, estos son asignados a las tareas mediante selección y arrastre del listado de roles a una tarea, permitiendo asignar tareas utilizando los colores de los usuarios como referencias.

Un ejemplo real donde esta aplicación se puede utilizar es en una reunión donde se debe analizar la implementación de una aplicación que automatice la recepción de órdenes de compra (ver Figura 22).

En esta reunión se realiza una primera aproximación al diseño del proceso, para lo cual el equipo de trabajo utiliza FlowChart Maker para establecer cómo sería el proceso, tres participantes en la reunión asume áreas de la empresa que participan de la orden de compra (ventas, servicio al consumidor y bodega), a la vez el participante actúa como defensor de su área evaluando la carga de trabajo o tareas que participa.

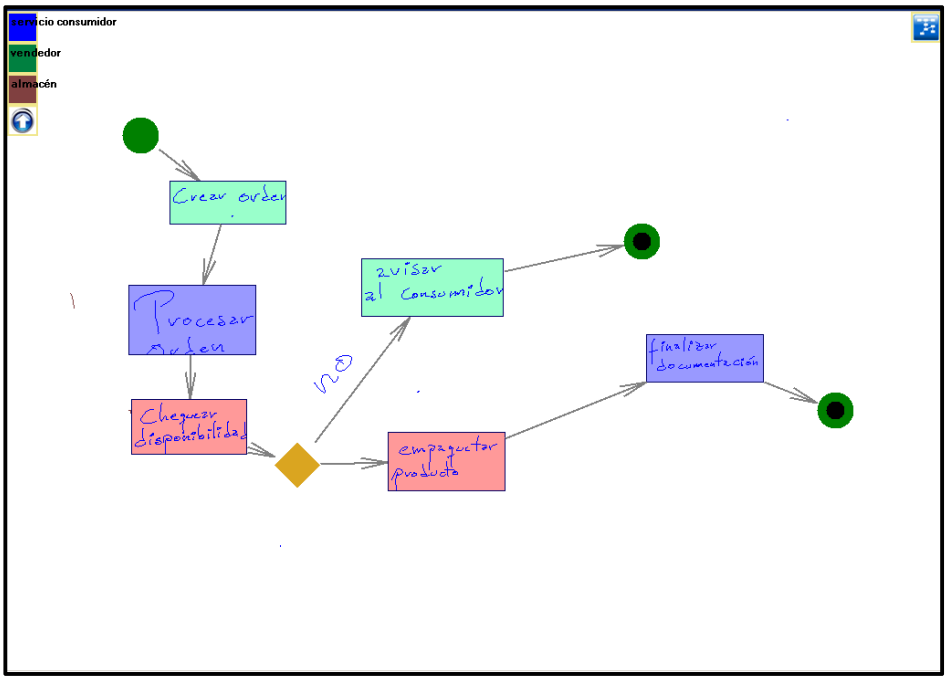
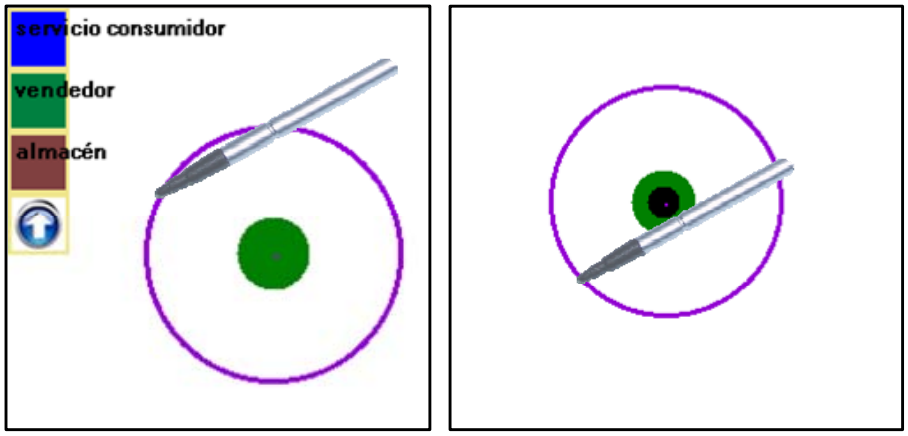


Figura 22: Diagrama generado de una orden de compra utilizando FlowChart Maker

5.1.1.1 Crear Grafos

La aplicación permite realizar los nodos más comunes de la notación BPM utilizando gestos.

Para realizar inicio y término de proceso se utiliza el gesto de círculo (CirculeGesture) y dependiendo de la dirección del movimiento se interpreta como uno u otro nodo.



(a) (b)

Figura 23: (a) ejemplo de creación de nodo de inicio, (b) ejemplo de creación de nodo de término.

También existen nodos que permiten realizar decisiones, en BPMN las decisiones se simbolizan con un rombo o diamante, pero cada tipo de decisión incluye un símbolo diferente dentro de este rombo, en NetSketcher se opto por los tipos de decisiones más comunes XOR, OR y AND, cada una de estas se generan con el gesto de triangulo (TriangleGesture) y según la orientación del triángulo se determina qué tipo de decisión se va a crear.

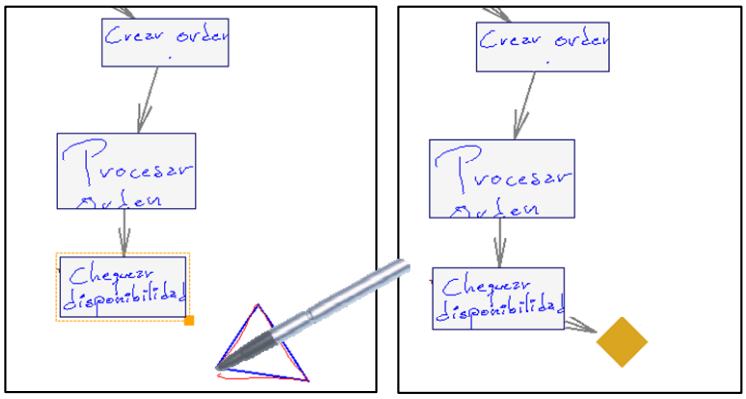


Figura 24: Ejemplo de creación de nodo de decisión, (a) se realiza el gesto TriangleGesture formado un triángulo con base horizontal, (b) se obtiene un nodo de decisión XOR.

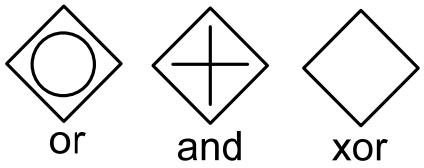


Figura 25: símbolos BPMN de nodos de decisión utilizados en FlowChart Maker.

Los nodos más importantes en BPMN son las tareas, estas indican una actividad atómica dentro del proceso, en FlowChart Maker, las tareas se generan realizando el gesto de ángulo recto (RectGesture) en dirección de izquierda a derecha.

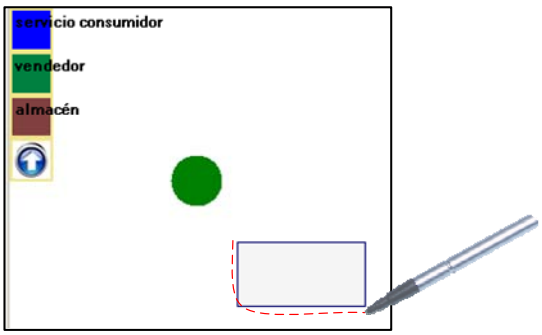
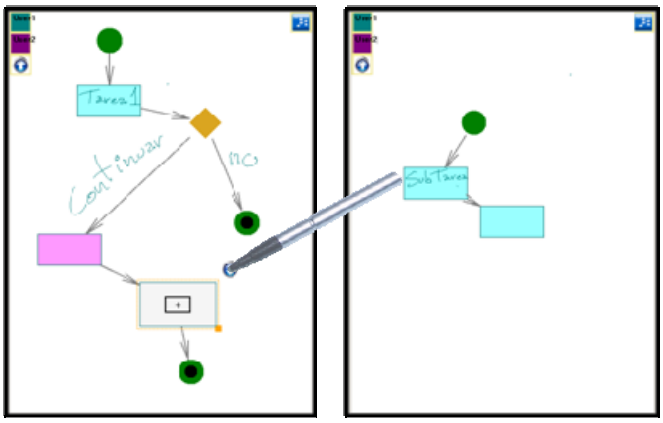


Figura 26: ejemplo de creación de un nodo tarea.

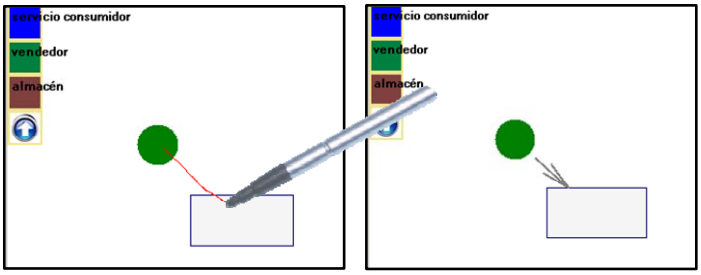
Debido al que los grafos pueden utilizar más espacio del que el canvas ofrece, FlowChart Maker ofrece la posibilidad de crear sub procesos, para realizar un nodo de sub proceso se utiliza el gesto RectGesture con el movimiento de derecha a izquierda, luego cuando se desea editar el sub proceso basta con realizar doble clic en el ícono superior derecho que aparece cuando se selecciona el nodo.



(a) (b)

Figura 27: Ejemplo de edición de un nodo de sub proceso, (a) doble clic en el ícono para entrar, (b) creación de sub proceso.

Para enlazar los nodos entre si se debe trazar una línea entre los 2 nodos, estas relaciones están limitadas de acuerdo al tipo de nodo que se crea, por ejemplo los nodos de tarea pueden enlazar solo con un nodo, pero puede recibir tantos enlaces como se deseen, estas pequeñas restricciones son útiles para mantener el orden del grafo.



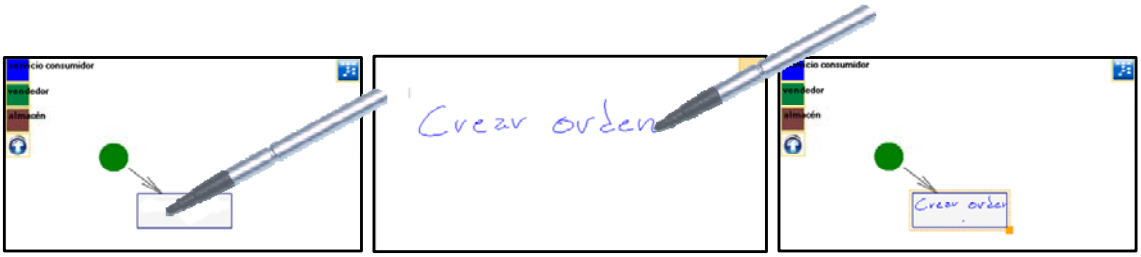
(a) (b)

Figura 28: Ejemplo de creación de un enlace de flujo, (a) unión de 2 nodos en la direccion del flujo del proceso, (b) ejemplo de enlace creado.

5.1.1.2 Edición de nodos

Luego de crear un nodo, usando el gesto de doble clic se puede *"entrar"* a este para realizar anotaciones, esto permite tener toda la pantalla para realizar anotaciones. Para volver al grafo se realiza el gesto de doble clic en algún lugar del espacio vacío

o a través del menú, una vez en el grafo se puede apreciar que el nodo con anotaciones muestra las anotaciones escritas anteriormente.



(a) (b) (c)

Figura 29: Ejemplo de edición de tareas, (a) doble clic en la tarea, (b) dentro de la tarea se hacen las anotaciones, (c) en el grafo las anotaciones de una tarea se ven como vista previa.

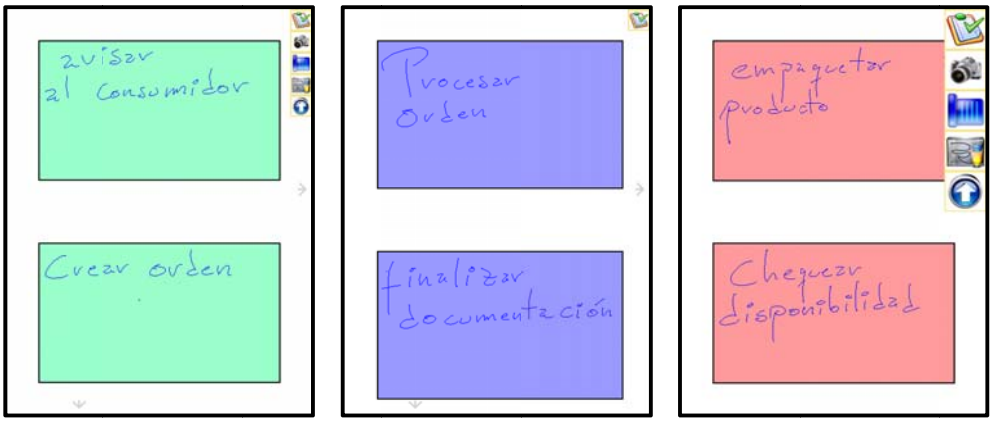
Para complementar la información del grafo también es posible realizar anotaciones sobre este, un ejemplo común es cuando se desea aclarar que significa tomar una u otra opción luego de pasar por un nodo de decisión.

5.1.1.3 Grafos anidados

El nodo de grafos anidados además de permitir anotaciones como en los nodos de tarea, también tiene la propiedad de realizar grafo en un sub espacio o Page, para ello se selecciona el nodo y aparece un botón en el borde superior derecho, haciendo doble clic sobre este botón se entra a un nuevo espacio, en este Page se puede realizar todo un nuevo grafo, exceptuando mas nodos con grafos anidados (Ver Figura 27).

5.1.1.4 Vista de tareas del usuario

A medida que el grafo crece se dificulta visualizar todas las tareas asignadas a un usuario, más aun, debido a los sub procesos que se puedan generar no es posible ver todas las tareas en una misma vista. Para solucionar este problema la aplicación provee de un modo que muestra el listado de tareas que le toca al usuario, de esta forma, aun si el grafo es complejo de ver, debido a su tamaño o a la cantidad de grafos anidados, este modo proporciona una visión completa de las tareas asignadas, con ello un usuario pude llevar la cuenta de cuantas tareas le toca o puede editar directamente la tarea pasando al modo de edición con doble clic sobre la tarea.



(a) (b) (c)

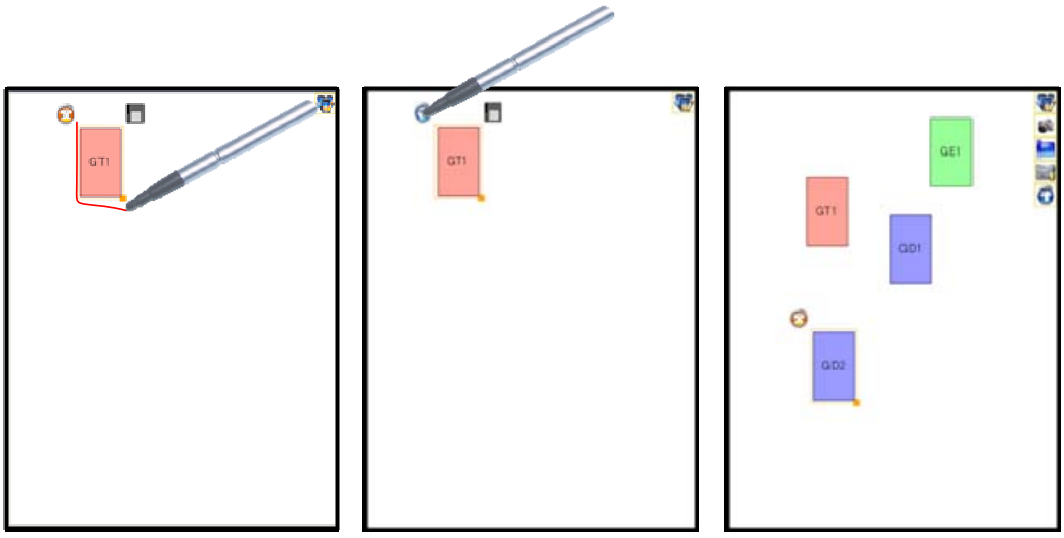
Figura 30: Vista de las tareas que tienen 3 participantes en un proceso, (a) tareas del área de ventas, (b) tareas del departamento de servicio al consumidor, (c) tareas de bodega.

5.1.2 GroupMode

Esta aplicación ejemplifica cómo se puede realizar una herramienta de creación y administración de grupos, GroupMode utiliza la mayoría de las propiedades de la clase Session y su característica principal es la utilización de objetos gráficos simbólicos para administrar los grupos.

GroupMode permite crear grupos, cambiar roles, realizar peticiones y respuestas hacia y desde el administrador respectivamente. Además GroupMode permite realizar más de un grupo a la vez por usuario, por supuesto todos los usuarios están habilitados para crear grupos.

Una posible situación donde se puede utilizar este sistema es en una sala de clase, donde el profesor a dado una tarea, la cual contiene varios puntos que se deben realizar, el profesor determina que los alumnos pueden elegir en qué puntos de la tarea desean trabajar, utilizando GroupMode el profesor crea varios grupos en el sistema de tal forma que los alumnos pueden decidir en qué punto de la tarea trabajar enviándole una petición al profesor, adicionalmente los alumnos que demuestran interés en trabajar en más de un tema, pueden realizar una petición para participar en más de un punto de la tarea o en caso que desean participar como observador del trabajo en otros grupos, pueden enviar una petición de invitado. Con este sistema los alumnos colaboran en el trabajo y permite que alumnos más participativos o están más avanzados, trabajen en más puntos de las tareas y alumnos más tímidos o menos participativos observar el trabajo de otros grupos.



(a) (b) (c)

Figura 31: (a) creación de un grupo, (b) activación de un nodo, (c) múltiples grupos creados por distintos usuarios.

5.1.2.1 Creación y Activación de un grupo

En GroupMode el único rol que está fijo es el de administrador, este rol se obtiene al crear un grupo. Un usuario puede crear un grupo al realizar el gesto de ángulo recto (RectGesture) sobre el área de trabajo de GroupMode, luego de crear un grupo, cualquier usuario puede acceder a este seleccionándolo y haciendo doble clic en el ícono superior izquierdo del menú de borde del grupo (Ver Figura 31 a y b).

Entrar a un grupo significa que todo lo que se cree en otras aplicaciones podrá ser visto y editado, por supuesto solo si un usuario tiene permisos en el grupo. Los objetos de distintos grupos, pero mismo modo, *"viven"* en el mismo espacio de trabajo, aun así no interactúan entre sí, debido a que sus propiedades de edición y visibilidad se bloquean y no permiten ser presentados en pantalla, ni modificados en los repositorios.

Las restricciones y permisos en GroupMode son extensibles a todas las aplicaciones gracias a que las políticas de control, utilizadas en los grupos, se manejan a nivel de la plataforma y no en una aplicación específica.

5.1.2.2 Modificación de roles

En GroupMode se puede modificar los permisos para un usuario utilizado el cambio de roles. Esto se hace entrando en el objeto que representa al grupo usando doble clic.

En este espacio de trabajo se ven 3 áreas, cada una representativa de un rol, en el área inferior están todos los usuarios de la red que no han sido invitados a la sesión, por lo general todos los usuarios aparecen en este estado inicialmente.

(a) (b) (c)

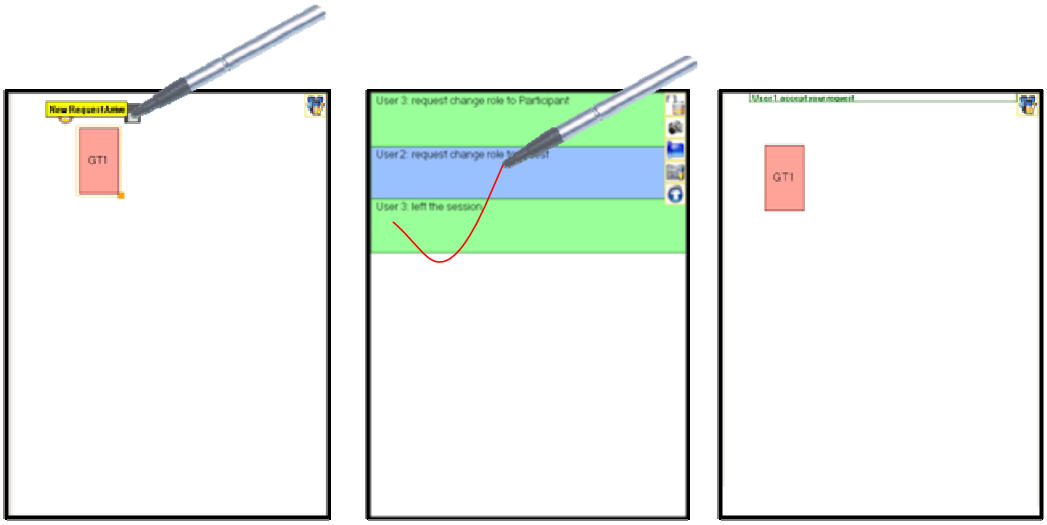
Figura 32: (a) doble clic para entrar a cambiar un rol, (b) espacio de modificación de roles, (c) desplazamiento de un usuario para cambiar un rol si es administrador o para realizar una petición si es otro usuario.

Para cambiar los permisos de edición o visibilidad de un usuario solo se debe arrastrar, su ícono representativo, hacia las áreas superiores, a su vez un cambio de rol indica una forma de evaluación distinta de las políticas de acceso, las cuales son determinadas en el Right del grupo, es por esto que si se desea cambiar la forma de evaluar un rol se debe hacer un nuevo Right y cargarlo en el Session del grupo.

5.1.2.3 Peticiones de participantes de un grupo

En el caso de usuarios que desean entrar a un grupo o cambiar su propio estado, deben acceder al mismo tipo de espacio de trabajo utilizado por el administrador para cambiar roles, con la diferencia que solo pueden cambiar su propio estado. Cuando un usuario cambia su estado, se envía una petición al administrador de ese grupo, quien acepta o rechaza el requerimiento.

Un administrador es el único que tiene la facultad de acceder al listado de peticiones, para ello en la página donde son visibles todos los grupos, debe seleccionar el grupo y luego hacer doble clic en el botón de la esquina superior derecha (Ver Figura 33.a), a través de este evento ingresa al listado de peticiones, el administrador puede hacer un ticket si acepta la petición o una cruz si la niega, a medida que vaya realizando estas acciones las peticiones van desapareciendo del listado.



(a) (b) (c)

Figura 33: (a) aviso al administrador de nuevas peticiones , administrador hace doble clic en el ícono de acceso a las peticiones, (b) administrador acepta una petición, (c) usuario que hizo la petición recibe un aviso, en forma de alerta, con la decisión del administrador.

Cuando el administrador toma una decisión, el usuario que realizó la petición tendrá un awareness avisándole sí el administrador aceptó o no su petición, para el caso del administrador también existen awareness cuando llegan nuevas peticiones.

5.2 Herramientas de menú

La plataforma cuenta con un menú principal y un menú contextual que es común para todas las aplicaciones, estos menús tienen el objetivo de favorecer la usabilidad de las aplicaciones. A continuación se explican algunas de las herramientas que se ofrecen para los usuarios.

Screenshot: botón en la barra principal que permite capturas una imagen de pantalla, permitiendo guardar como un archivo tipo PNG el estado actual en la aplicación.
Paleta de colores: se ofrece al usuario una paleta de 4 colores basada en el color que seleccionó al inicio de la aplicación, optando por uno o por otro color puede

cambiar el color de los trazos que realiza, esta paleta de colores es útil para realizar trazos que contrasten con el fondo.
Guardar y Cargar sesión: opciones del menú contextual que permiten almacenar todos los datos de la página actual y sus páginas anidadas en un archivo, gracias a estas acciones el usuario puede continuar con el trabajo adelante o compartir sus avances por medio del archivo generado.
Carga de Imágenes y textos: opciones del menú contextual que permiten cargar imágenes desde un archivo o texto ingresado por teclado.

5.3 Soporte para testeos y apoyo para desarrolladores

Un aspecto complementario para que una plataforma sea útil es el soporte que se pueda ofrecer a los desarrolladores, debido que es una plataforma pensada para desarrollar diversas aplicaciones y extensiones de ella misma. Esta sección explica las herramientas que se proveen para fomentan la utilización y el crecimiento de la plataforma.

5.3.1 Documentación

La documentación está considerada como parte de los requerimientos de este proyecto, ya que permite reforzar la comprensión de la plataforma y entregar esquemas para el trabajo sobre esta.

Se realizó una documentación de la API que contiene la descripción de métodos y clases que conforman la plataforma, detallando su funcionalidad de manera concisa, con la finalidad de apoyar el trabajo diario de los programadores de aplicaciones.

También se incluye una guía descriptiva para realizar una aplicación, explicando cómo funcionan los Modes, Gestures y Sketches, de tal forma que cualquier programador que se inicie en el trabajo sobre la plataforma pueda entender cómo realizar una aplicación y aproveche al máximo el potencial que le entrega la plataforma.

5.3.2 Aplicaciones y herramientas para el testeo

- **Bitácora:** La clase Log es una herramienta que registra la bitácora de una aplicación, esta clase está incluida de forma automática en el proyecto, todos los errores que suceden en la plataforma son capturados y enviados al log, detallando el tiempo, tipo de error y su ubicación en el código, esta herramienta es muy utilizada en testeos entregando información que luego puede ser analizada.

En el log está considerado el posible acceso concurrente a este y almacena en memoria secundaria los registros cada vez que una nueva línea es agregada. Además mantiene una lista en memoria primaria las últimas "*n*" líneas que se han agregado, permitiendo tener una vista previa del log en tiempo de ejecución.

```
public void SetLevel(logLevel level,int buffer)

public void Write(String info, logLevel level)
```

Código 16: métodos más utilizados de la clase Log, con el método SetLevel se puede filtrar que mensajes se almacenaran y cuantas líneas se mantendrán disponibles en memoria RAM.

- **Reportes estadísticos:** La clase StatisticsReport está pensada para ser heredada y construir un generador de tablas de datos, al igual que Log prevé el uso concurrente y almacena la información en memoria secundaria, StatisticsReport se encarga de crear archivos tipo CSV para mostrar listados de datos.

  Para utilizar este tipo de reportes StatisticsReport cuenta con un constructor que da la estructura de cómo y cuantos datos se recolectaran, un método que permite ingresar una fila de datos y un método para terminar la recolección de datos.

```
public StatisticsReport(string nameFile, int cols,
List<string> headers)
```

Código 17: constructor de la clase StatisticsReport, se requiere un nombre, la cantidad de columnas y las etiquetas de cabecera de estas columnas.

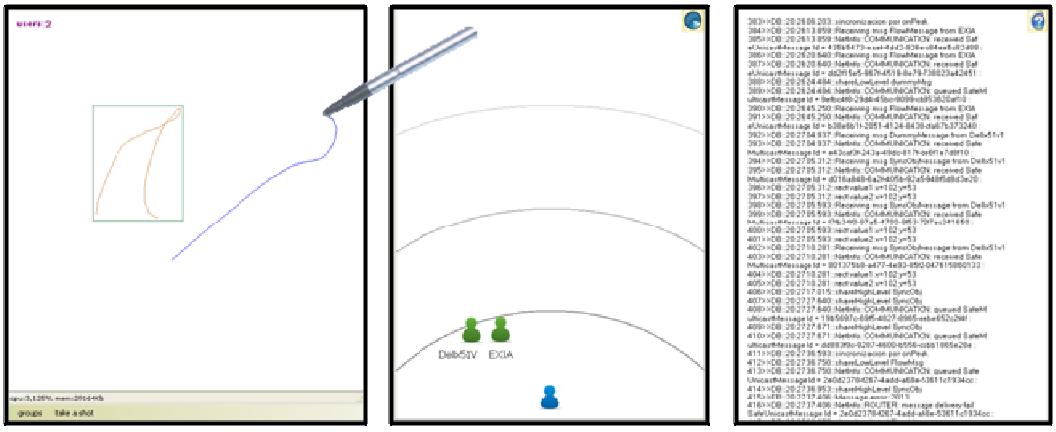
StatisticsReport es una herramienta útil para recabar información y luego generar tablas y gráficos con los cuales dar sentido a los datos. Un caso aplicado de StatisticsReport son las clases TrafficReport, utilizado para analizar el tráfico de mensajes en la red, y PerformanceReporte, reporte de consumo de recursos (Memoria y CPU), los 2 reportes son utilizados en la aplicación Drawing Strokes para estudiar el comportamiento de la capa de sesión.

- **RadarMode:** este Mode se utiliza como indicador de estado, calidad de señal y distancia en *"saltos"* (Hops) de los usuarios en la sesión. RadarMode entrega una vista rápida a la red, útil para verificar la calidad de la red y la cantidad de usuarios con los que se interactúa.

- **DebuggingMode:** Mode que visualiza el Log en tiempo de ejecución, de esta forma no es necesario esperar a terminar una sesión para verificar los errores. DebuggingMode utiliza el cache de la clase Log para mostrar las últimas líneas del Log, además permite navegar en el log y se ajusta al tamaño de la pantalla facilitando su revisión.

- **Drawing Board:** Modo que implementa todos los gestos, permite realizar pruebas de los gestos y nuevas funcionalidades de la plataforma, es la aplicación demo de la capa de espacio de trabajo, en esta se pueden crear los 4 tipos de Sketch, funciona colaborativamente y cuenta con acciones para la mayoría de los gesto de la plataforma.

- **Drawing Strokes:** debido a que la plataforma puede ser utilizada a nivel de la capa de sesión, también existe una aplicación de testeo para esta capa, con esta aplicación es posible dibujar trazos, seleccionarlos y moverlos, con ello se pueden probar la creación y modificación de objetos, además cuenta con opciones de menú para crear sesiones entre 2 usuarios y entrega estadísticas de consumo de CPU y RAM (solo para la versión PC).



(a) (b) (c)

Figura 34: (a) aplicación Drawing Strokes, (b) modo de NetSketcher RadarMode ,(c) modo de NetSketcher DebuggingMode.

6 Evaluación de la plataforma

Evaluar la calidad de la plataforma es bastante complejo, ya que decir que es eficiente o no, depende mucho con que se comparé, igualmente sucede al tratar de evaluar la efectividad en los tiempos de desarrollo.

En cuanto a la red, se realizaron pruebas entre 3 a 7 equipos durante todo el proceso de desarrollo, en las pruebas participaron alumnos que utilizaban la aplicación Drawing Board para interactuar, se utilizaron los reportes generados por la clase Log para revisar problemas específicos y TrafficReport para chequear la frecuencia y tipos de mensajes enviados. Muchas de las fallas se evidenciaron cuando aumentaba la cantidad de equipos de la red y fueron útiles para afinar desperfectos.

Como resultado de las pruebas de red se logró la respuesta en red de 2 segundos durante una sesión. Se encontró que cuando un usuario escribe a mano alzada, la transmisión de información utiliza las estrategias de envió diferenciado y homogenización de tamaño de paquetes, esto debido a que la velocidad de escritura hace que la plataforma espere para sincronizar varios trazos a la vez, este es un ejemplo que respalda el uso de las estrategias utilizadas para optimizar la transmisión de información.

Otro tema evaluado es la complejidad en programación sobre la plataforma, se entregó la plataforma a 2 alumnos de computación de 3er año, se les dio como tarea realizar una aplicación sobre la plataforma, a los alumnos se les entrenó en el uso de la plataforma realizando 4 talleres, explicando desde como instalar y poner en funcionamiento un proyecto en Visual Studio 2008, hasta como realizar administración de grupos, todo esto apoyado con un foro para resolver sus dudas sobre la plataforma. A medida que los alumnos trabajaban sus comentarios y dudas permitían comprender qué tan complejo resultaba programa, testear y utilizar las aplicaciones.

Tan solo a las 3 semanas después del segundo taller, los alumnos crearon su primer gesto y modo. Esto es una evidencia de la sencillez del trabajo en la plataforma, considerando que estos alumnos nunca habían trabajado en C# antes de esto.

7 Conclusiones

La plataforma logró sus objetivos de compatibilizar la computación móvil con el trabajo colaborativo, el sistema contó con ejemplos claros de áreas y temas donde una aplicación de este estilo resulta más útil que los formatos usados actualmente. La plataforma ofrece nuevas propuestas a los problemas de administración de la información colaborativa y cohesiona distintas propuestas ofrecidas en otras aplicaciones colaborativas de manera harmónica y funcional.

Quizás el punto pendiente es la entrega de resultados medibles en las pruebas realizadas, aun así hay que tener presente que durante todo el proyecto la plataforma estuvo *"expuesta"* a evaluación por profesores y alumnos del área de computación, ellos probaron las aplicaciones e implementaron sobre esta, toda la información que ellos aportaron no es fácil de cuantificar.

En cuanto al desempeño de las aplicaciones sobre la plataforma, estas se comportan de manera aceptable de acuerdo a los requerimientos propuestos en la sección 3, todos los equipos respondían bien a la carga, solo se encontró algo de lentitud en la captura de información desde pantalla en los equipos Dell, equipos que cuentan con menos memoria y capacidad de procesamiento que el resto.

Una de las características más relevantes del diseño de la plataforma es el repositorio dual, es propio de esta plataforma y gracias al concepto de minimizar los puntos de sincronización, permitió crear la estrategia de envió diferenciado de mensajes. Además su interfaz de manejo de objetos les permite a los programadores trabajar de forma transparente la manipulación de objetos compartidos. Fue una excelente forma de abordar el problema de administración de objetos.

La plataforma NetSketcher es una potente herramienta que puede ser utilizada para diversos proyectos a futuro, está pensada para seguir perfeccionándose y ser utilizada por alumnos del área de computación para realizar investigación o como herramienta de aprendizaje sobre sistemas colaborativos.

8 Trabajos futuros

8.1 Sesiones asincrónicas

Una propuesta que entregaría a la plataforma la propiedad de almacenar de forma centralizada una sesión y recuperarla más tarde es el trabajo colaborativo asíncrono, este tipo de sesiones existen en otras plataformas y permiten la interacción con otros a pesar de que no todos estén en el mismo momento, esto extiende la variedad de aplicaciones que se podrían crear.

Una de las dificultades que tiene es a nivel conceptual, ya que se necesita la mejor forma de compatibilizar una red distribuida con este sistema centralizado.

Para realizar sesiones asincrónicas se propone integrar un sistema de información de servicios que ofrezcan los dispositivos, en el caso particular de sesiones asíncronas una base de datos para persistir la sesión.

Una buena opción para realizar un sistema de persistencia en disco es utilizando SQLite, librería que permite integrar todas las propiedades que cuenta una base de datos, pero formando parte de la plataforma como una sola aplicación.

8.2 Pre-gesto y análisis de gestos

Actualmente se está implementando una aplicación que cuenta con un nuevo concepto, los pre-gestos (PreGesture), muchas veces un usuario cuando dibuja un trazo pretendiendo hacer un gesto termina realizando un trazo en pantalla. Por ejemplo, si el usuario desea hacer el gesto CirculeGesture, pero la plataforma interpreta esto como un círculo, entonces no se realiza el gesto que él deseaba.

Este problema es debido a que hacer un dibujo o realizar un gesto se confunden entre sí, para corregir esto están los pre-gestos, que le dicen a la plataforma si el usuario desea dibujar o hacer gestos.

Para realizar un gesto con pre-gesto la plataforma identifica al pre-gesto, si este sucede, lo que viene a continuación será un gesto, en caso de que no ocurra, entonces el usuario desea dibujar.

Pero existen variadas formas de hacer un pre-gesto y la idea detrás de la aplicación de pre-gesto es probar que opción es la mejor, es por eso que se crearon 3 formas de hacer un pre-gesto y se pretende entregar esto a un grupo de usuarios que prueben las opciones y den su opinión.

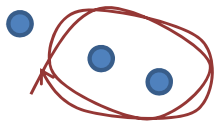
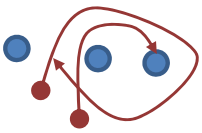
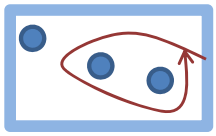
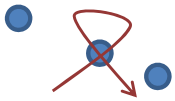
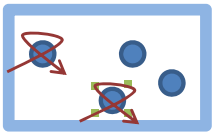
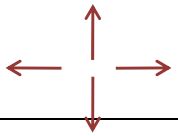
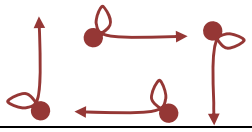
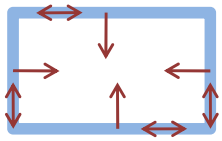
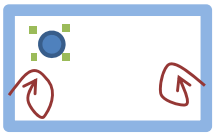
	Dependen de la repetición o la velocidad	Depende de una marca inicial (doble-clic o círculo)	Depende de una posición inicial absoluta
Selección múltiple			
Borrar			
Hacer /deshacer			
Scroll			
Copiar/Pegar			

Figura 35: Tabla con los pre-gestos que se están estudiando.

Actualmente se está diseñando el escenario para trabajará en las pruebas y seleccionado el grupo de personas que serán sujetos de prueba.

8.3 Motor de gestos

Se propone que a futuro de realice un motor de gestos que de una estructura general para realizar gestos. Los gestos son la parte más compleja al momento de realizar una aplicación, reconocer un gesto no es fácil, el error más común es el falso positivo, debido a que muchas veces no se desea hacer un gesto y este sucede de todas maneras, restringir los gestos reduciendo el rango de error que se acepta causa un problema: a medida que endurecemos las restricciones dificultamos la creación del gesto, sin embargo mejoramos las posibilidades de que no suceda un falso positivo. En el caso contrario, relajar las restricciones, facilita la creación del gesto, pero también los falso positivos.

Es importante realizar una investigación en este campo que de cómo resultado un motor de gestos que evalúe de manera general y más precisa los gestos de la plataforma, solucionando el problema de falsos positivo y simplificando la creación de gestos.

8.4 Librerías de comunicación

Se propone realizar una librería anexa a la plataforma que permita realizar comunicación a través de Bluetooth e IRDA, con ellas proveer mas formas de interacción entre los usuarios y optimizar la forma de intercambiar la información entre dispositivos.

Actualmente se han hecho pruebas de comunicación Bluetooth, con el fin de proveer redundancia en el intercambio de información entre dispositivos de la red. El objetivo es ofrecer más confiabilidad en la comunicación de la plataforma.

8.5 Habilitación en otros sistemas operativos

Dado la popularidad de los sistemas operativos Android de Google y iOS de Apple, se hace necesario evaluar la factibilidad de ofrecer operatividad de la plataforma en estos S.O.

La plataforma dado que está hecha en C#, es posible compilarla con algunos cambios en Linux usando el compilador de Mono, con ello sería posible utilizarla en dispositivos que usen Android.

El problema es que varias de las librerías que se utilizan en los sistemas operativos de Microsoft no están disponibles para los otros S.O., a pesar de todo, estas librerías solo son para mejorar el aspecto gráfico de las aplicaciones o realizan configuraciones de la tarjeta de red y todo esto puede ser reimplementado o descartado para versiones en otros S.O. sin perder las propiedades de la plataforma.

8.6 Manejo de políticas de acceso

El sistema de Right logra concentrar toda la lógica de grupos y ofrece herramientas para ser modificado, pero hasta el momento no se han estudiado escenarios donde se deban hacer nuevos Right.

Sería interesante proponer escenarios donde se deban implementar lógicas más complejas de políticas de control y ver hasta donde el sistema de Right es una forma de trabajar rápida, transparente y eficiente.

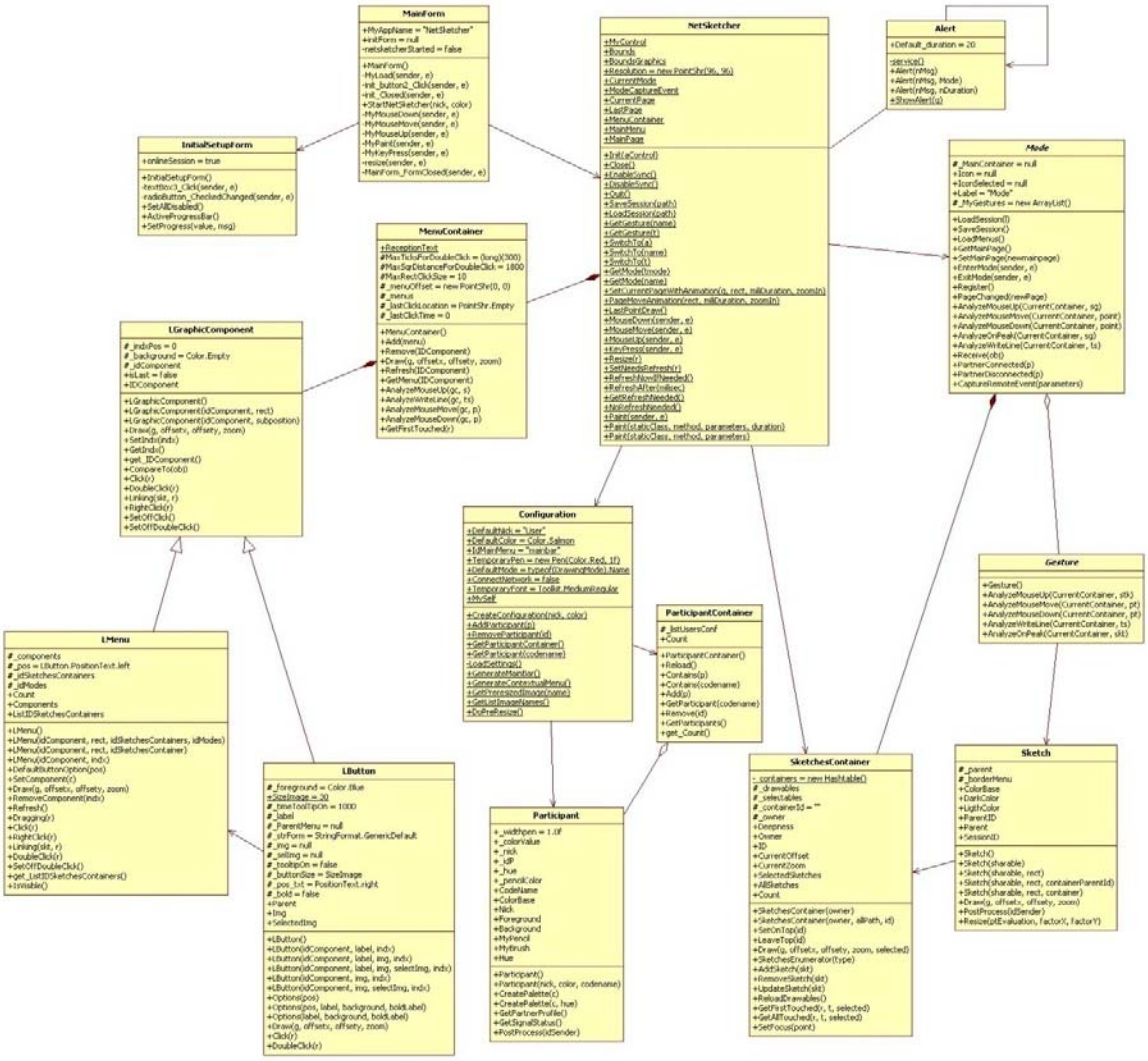
8.7 Estudiar los valores de sincronización

En todas las capas de la plataforma existen estrategias que optimizan la sincronización, por ejemplo, tratando de homogenizar el tamaño de los mensajes o tratando de anticiparse a los mensajes incompletos o erróneos que pueda realizar un usuario.

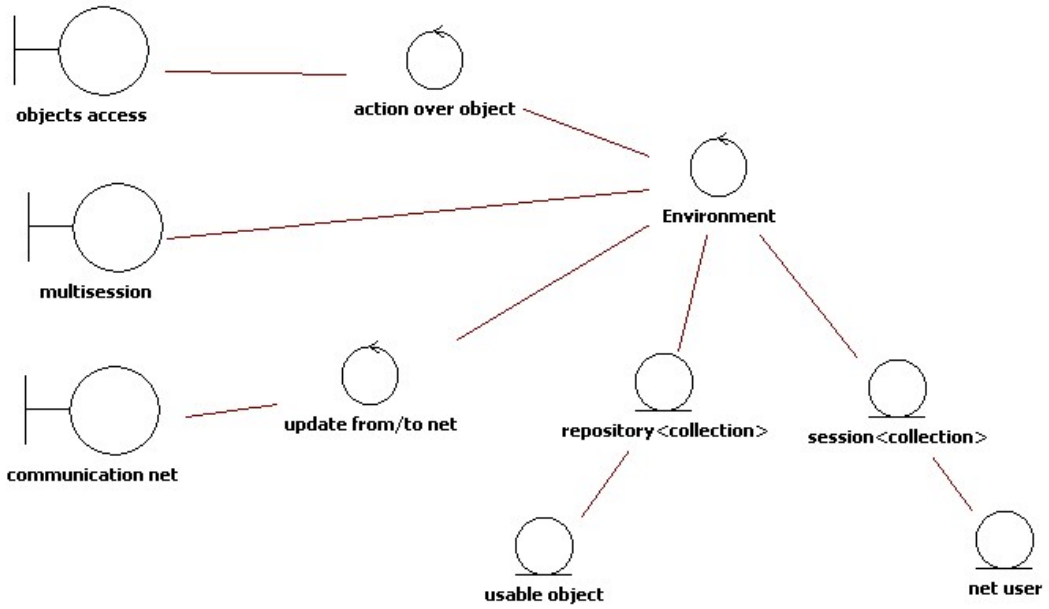
Todas estas estrategias utilizan variables que pueden ser ajustadas, para ello se requiere realizar pruebas con distintos valores y de los resultados ver que combinación de valores es más eficiente.

9 Anexos

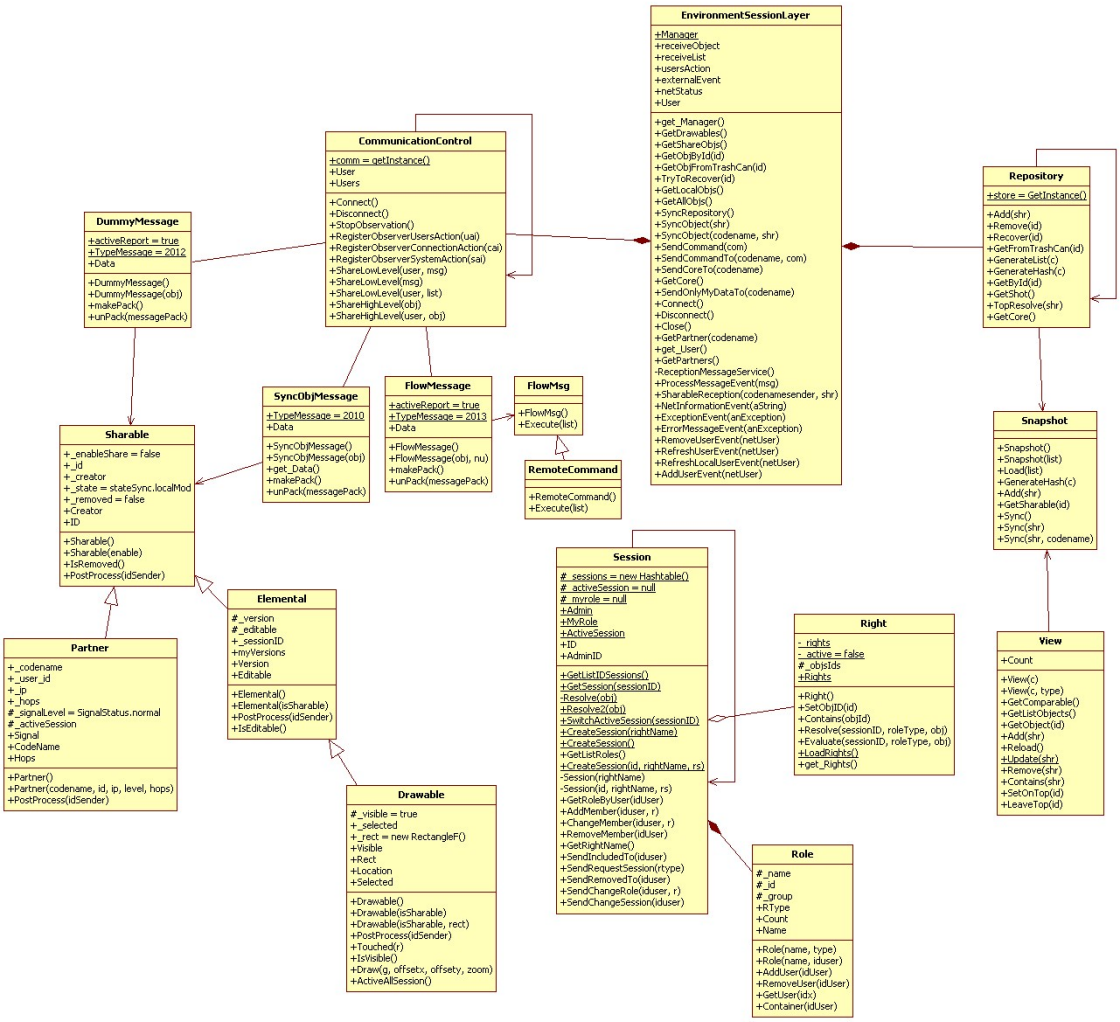
Anexo A: Diagrama de clases Capa de Espacio de Trabajo



Anexo B: Modelo de análisis de la Capa de Administración de Sesión



Anexo C: Diagrama de clases de la Capa de Administración de Sesión.



10 Bibliografía

[1] MAUTHER, A ., HUTCHISON,D. Peer-to-Peer Computing: System, Concept and characteristics. Lancaster University, Lancaster, Inglaterra. pp 1, 4-8. 2003.

[2] MATTERN, F. Wireless Future: Ubiquitous Computing. ETH Zürich. Pp 1,2. 2004.

[3] EL SADDIK, A., RAHMAN, Md. A., HOSSAIN, M. A. Authoring Multimedia Object in Collaborative Ambient Intelligent Virtual Environment. Multimedia Communications Research Laboratory, School of Information Technology and Engineering (SITE), University of Ottawa, Ottawa, Ontario, Canada. pp. 160. 2005.

[4] EL SADDIK, A., RAHMAN, Md. A., ABDALA, S., SOLOMON, B. PECOLE: P2P multimedia collaborative environment. Multimedia Communications Research Laboratory, University of Ottawa, Ottawa, Ontario, Canada. pp. 356-358. 2008.

[5] GUERRERO, L. A., PORTUGAL, R. C., FULLER, D. A. TOP: A Platform form Development of Web Interfaces and Collaborative Applications. Departamento de Ciencias de la Computación, Facultad de Ciencias Físicas y Matemáticas Universidad de Chile, Santiago, Chile. pp 1-3. 1999.

[6] RODDEN, T. A technological framework for CSCW. CSCW: Some Fundamental Issues, IEE Colloquium on, London, UK. pp 7/1. 1991.

[7] BALOIAN, N, LUTHER, SÖFKER, URANO. Developing Mobile Collaborative Applications. Editorial: Multimodal Human-Machine interaction in Different Application Scenarios. Logos Verlag, Berlin, Germany. pp. 31-55. 2008.

[8] ELLIS, C.A., GIBBS, S. J. Concurrency Control in Groupware Systems. MCC, Austin, Texas, USA. pp 399. 1989.

[9] LUKOSCH S., UNGER C. Flexible Management of Shared Groupware Objects. University of Hagen, Department for Computer Science, Alemania .pp 9. 2000.

[10] SCHUCKMANN, C., KIRCHNER, L., SCHÜMMER, J., HAAKE, J. M. Designing object-oriented synchronous groupware with COAST. IPSI - Integrated Publication and Information Systems Institute GMD - German National Research Center for Information Technology Dolivostr, Alemania .pp 1, 7. 1996.

[11] NEYEM, A., OCHOA, S. F., PINO, J. A., GUERRERO, L. A. Sharing Information Resources in Mobile Ad-hoc Networks. Department of Computer Science, Universidad de Chile, Santiago, Chile. Pp 352-257. 2005.

[12] BUSZKO, D., LEE, W-H., HELAL, A. Decentralized Ad-Hoc Groupware API and Framework for Mobile Collaboration. Motorola iDEN Group, Florida , USA. 2001.

[13] PENDERGAST, M. O. A Comparative Analysis of Groupware Application Protocols. Washington State University, Washington, USA. pp 2,3,10. 1998.

[14] GUERRERO, L. A., FULLER, D. A. A pattern system for the development of collaborative applications. Depto. de Ciencias de la Computación, Universidad de Chile y Depto. de Ciencias de la Computación, Ponticia Universidad Católica de Chile, Santiago, Chile. 2001.

[15] CEGLAR, A., CALDAR, P.. A New Approach to Collaborative Frameworks using Shared Objects. Flinders University of South Australia School of Informatics and Engineering, Australia. 2001.

[16] EDWARDS, W. K. Policies and Roles in Collaborative Applications. Xerox Palo Alto Research Center, Palo Alto, California, USA. 1996.

[17] CHABERT, A., GROSSMAN E., JACKSON, L., PIETROWICZ S., SEGUIN, C. Java Object-Sharing in HABANERO. pp 1,2.1998.

[18] RODRIGUEZ-COVILI, J., OCHOA, S., PINO, J. HLMP: High Level MANET Protocol. (CSCWD'10), Shanghai, China. 2010.

[19] RODRIGUEZ-COVILI, J., OCHOA, S., PINO, J., MESSEGUER, R., MEDINA, E., ROYO, D., HLMP API: A Software Library to Support the Development of Mobile Collaborative Applications. Departamento de Ciencias de la Computación, Universidad de Chile, Chile. pp 479 – 481. 2010.

[20] SEET, B-C. Mobile Peer-to-Peer Computing for Next Generation Distributed Environments. pp 421- 424, 432-434, 438. 2009.

[21] KOTILAINEN, N. Mobile Chedar - A Peer-to-Peer Middleware for Mobile Devices. 2005.

[22] LANDAY, J. A., MYERS, B. A., Extending An Existing User Interface Toolkit To Support Gesture Recognition, School of Computer Science Carnegie Mellon University,Pittsburgh, USA. 1993.

[23] KORTUEM, G. Proem: A Middleware Platform for Mobile Peer-to-Peer Computing. 2002.

[24] KORTUEM, G. Proem: A Peer-to-Peer Computing Platform for Mobile Ad-hoc Networks. pp 1-3, 6. 2002.

[25] BISIGNANO, M. JMobiPeer: a middleware for mobile peer-to-peer computing in MANETs. pp 2-4, 2006.

[26] ROSEMAN, M., GREENBERG, S., GROUPKIT A Groupware Toolkit for Building Real-Time Conferencing Application. Department of Computer Science University of Calgary Calgary, Alberta, Canada. pp 3. 1992.

[27] ROSEMAN, M., GREENBERG, S., Building Real Time Groupware with GroupKit, A Groupware Toolkit. Department of Computer Science University of Calgary Calgary, Alberta, Canada. pp 9. 1995.

[28] GONG, L., Project JXTA: A Technology Overview. Sun Microsystems, Inc., Palo Alto, CA., USA. 2002.

[29] MYERS, B., HUDSON, S. E., PAUSCH, R., Past, Present, and Future of User Interface Software Tools. Carnegie Mellon University. pp 4-20. 2000.

.