Universidad Autónoma Metropolitana

Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Ingeniería en Computación

REMODELACIÓN PLÁSTICA DE LA BARRA DE HERRAMIENTAS DE UN PIZARRÓN COMPARTIDO.

Alumno:
Juárez Cerón Hector - 206305086

Trimestre Lectivo 13-O

Asesor del proyecto:
Dra. María Lizbeth Gallardo López

Resumen

El trabajo en equipo promueve la ayuda mutua entre sus participantes, por esta razón, las tecnologías de la información se han orientado a proporcionar herramientas de cómputo que faciliten el trabajo en equipo, como por ejemplo el denominado `CSCW` (*Computer-Supported Cooperative Work*) el cual permite a los colaboradores el poder participar cooperativamente en la producción de entidades compartidas e intercambiar mensajes para coordinarse, aun cuando ellos no puedan reunirse físicamente en el mismo lugar y/o al mismo tiempo, por ejemplo editores cooperativos de documentos compartidos y distribuidos en internet.

Además, el uso de dispositivos móviles, hacen más prácticas y dinámicas las actividades de personas que los utilizan. La aplicación `pizarrón compartido`, es un prototipo enfocado a fusionar el `CSCW` y el área de `HCI` (*Human-Computer Interaction*). Esta aplicación actualmente no puede adaptarse a los distintos dispositivos que la contienen, es decir no cuenta con la propiedad de plasticidad. La plasticidad denota la capacidad de la Interfaz gráfica de Usuario también conocida como GUI (*Graphical User Interface*) para soportar variaciones en el contexto de uso, preservando su usabilidad. Es decir una `GUI` plástica preserva la usabilidad si sus propiedades, seleccionadas durante el diseñó, se mantienen dentro un rango de valores cuando ocurren cambios contextuales.

Índice general

Capítulo 1

Introducción y Objetivos

1.1. Introducción

Un Pizarrón Compartido será una aplicación de dibujo parecida al Paintbrush de Microsoft donde varios usuarios podrán trabajar sobre la edición de un dibujo (al cual denominamos objeto); y cada uno de los usuarios podrá estar conectado a la aplicación a través de distintos dispositivos, a saber: PC, lap-top, ó dispositivos móviles. Un prototipo de esta aplicación fue desarrollado durante la tesis de maestría de Gabriela Sánchez Morales en el Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional (CINVESTAV), el título de la tesis es: *Redistribución semi-plástica mixta: el caso de estudio de un pizarrón compartido"*[2]. Esta tesis se sitúa en la intersección de dos dominios de investigación: Trabajo Cooperativo Asistido por Computadoras (CSCW) por sus siglas en inglés: *Computer-Supported Cooperative Work e Interacción Ser Humano-Computadora* (HCI) por sus siglas en inglés: *Human-Computer Interaction.*

El modelo de la aplicación Pizarrón Compartido tiene 5 interactores los cuales son: 1) área de dibujo, 2) barra de colaboradores, 3) meta-interfaz de distribución, 4) la ventana de autenticación y 5) la barra de herramientas. Este último interactor es el motivo de estudio de este proyecto terminal, el cual tiene como objetivo desarrollar una barra de herramientas plástica capaz de adaptarse a cualquier dispositivo. La plasticidad de una interfaz gráfica de usuario es una propiedad que poseen algunas aplicaciones de cómputo interactivas mono-usuario. Este concepto se inspira de algunos materiales que puede expandirse y contraerse bajo restricciones naturales, sin romperse. Aplicada a interfaces graficas de usuario, la plasticidad es la capacidad de adaptarse a cambios en el contexto de uso, preservando la "usabilidad"[1]. Una propiedad ineludible de la plasticidad es la "garantía de continuidad" cuando se produce una migración de la aplicación de un contexto de uso a otro. Una interfaz de usuario plástica preserva la usabilidad si sus propiedades: eficiencia, eficacia, entre otras, especificadas durante el diseño, se mantienen dentro de un rango de valores cuando ocurre la adaptación a cambios contextuales.

1.2. Motivación

El diseño y el desarrollo de interfaces de usuario plásticas constituyen una dirección de investigación prometedora: La plasticidad de una interfaz de usuario le permite adaptarse dinámicamente a las características del entorno actual, de la plataforma y/o de los dispositivos de trabajo. Un elemento importante de la aplicación pizarrón compartido es la barra herramientas, la cual deberá mostrarse en cualquier dispositivo que empleé el usuario; esto implica que la barra de herramientas deberá cambiar su presentación (remodelarse), en función del área de despliegue del dispositivo. El objetivo de este proyecto consiste en remodelar una barra de herramientas, siguiendo dos estrategias:

- 1. División por espacios de trabajo; es decir por tareas lógicamente conectadas.
- 2. Gestión de interactores o widgets; por ejemplo, ocultando los widgets menos importantes.

1.3. Objetivos del proyecto

El objetivo principal del proyecto terminal fue:

Diseñar e implementar un algoritmo que permita desplegar dinámicamente la barra de herramientas de un pizarrón compartido, en función del área del dispositivo de cómputo que lo despliegue.

Objetivos particulares del proyecto:

- **Realizar pruebas de usabilidad sobre tamaños y posiciones de los diferentes interactores que conforman la barra de herramientas.** Se realizó un estudio de los diferentes tipos de dispositivos que existen el mercado para determinar sus capacidades como son: capacidad en hardware (tamaño y resolución de pantallas), versiones de sistemas operativos disponibles para PC y dispositivos móviles.
- **Diseñar e implementar un algoritmo que permita desplegar dinámicamente la barra de herramientas, considerando el tamaño del dispositivo.** Para la realización del algoritmo se consideraron las resoluciones más usadas actualmente con la cuales se determinó el tamaño y ubicación de los interactores, así como el concepto de plasticidad. Se implementaron las clases que permiten la agrupación de los interactores sin perder su usabilidad.
- **Realizar pruebas para determinar la ubicación del algoritmo (cliente-servidor versus peer to peer).** No fue posible realizar completamente este objetivo porque la comunicación entre los dispositivos no fue un objetivo del proyecto; sin embargo la aplicación por su diseño modular cuenta con la capacidad de integrarse con un módulo de comunicación tipo cliente-servidor o peer to peer.

- **Diseñar e implementar una segunda etapa del algoritmo para medir la frecuencia de uso que, sobre los interactores, realizó el usuario en la última sesión; además de medir la frecuencia de uso, que sobre los interactores realiza el usuario en la sesión actual.** Se desarrolló una clase la cual permite leer las interacciones que el usuario realizó con cada interactor en su sesión anterior con la cual ordena los interactores de acuerdo al más usado.
- **Migrar el algoritmo a la aplicación de pizarrón compartido.** No fue posible realizar este objetivo sin embargo, el diseño modular de la aplicación permite ser integrada con la aplicación del pizarrón compartido creando una instancia de la barra de herramientas.

1.4. Organización del Reporte

El Documento actual se organiza en 5 capítulos ver figura 1.1.

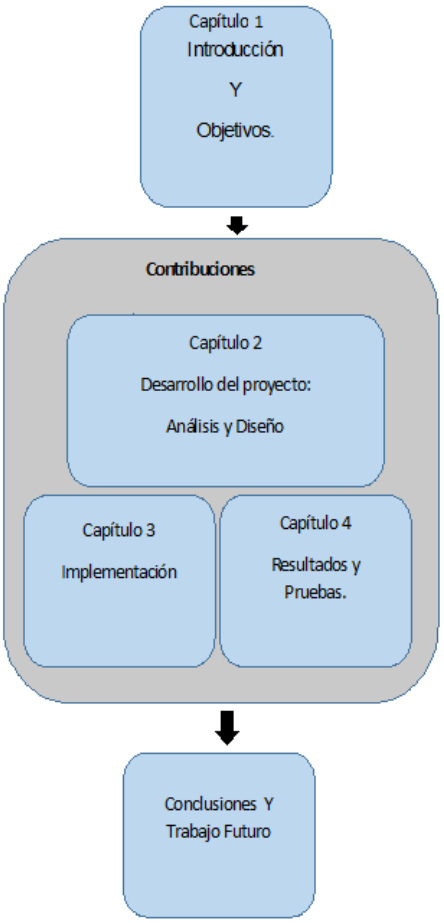

Figura 1.1: Organización del Reporte.

- **El capítulo 1:** Contiene la introducción, así como la descripción de los objetivos del proyecto.
- **El capítulo 2:** Contiene la explicación del diseño de la barra de herramientas.
- **El capítulo 3:** Contiene la descripción de la implementadoción de la barra de herramientas.
- **El capítulo 4:** Contiene la explicación de los resultados y pruebas realizadas para validar la barra de herramientas.
- **El capítulo 5:** Finalmente se presentan las conclusiones del proyecto, así como el trabajo futuro a realizar con la barra de herramientas.

Capítulo 2

Desarrollo del proyecto: Análisis y Diseño

El desarrollo del proyecto *Remodelación plástica de la barra de herramientas de un pizarrón compartido* fue desarrollado en tres etapas: Análisis, diseño e implementación, se realizó en dos versiones: para PC y para teféfono celular con sistema operativo Android. La metodología de desarrollo empleada fue el proceso unificado (UP); la implementación se realizó con los lenguajes de programación java en su versión JSE para PC y Android para dispositivo móvil.

2.1. Metodología de base empleada en el proyecto.

El desarrollo del proyecto se basó en la metodología de desarrollo de software UP la cual constituye un estándar para el análisis, implementación y documentación de sistemas orientados a objetos. La metodología cuenta con 4 fases: inicio, elaboración, construcción y transición. La metodología se caracteriza por ser iterativa e incremental; es decir, en cada iteración se busca mejorar un componente o módulo del sistema; y cada componente que se desarrolla corresponde a un incremento del sistema.

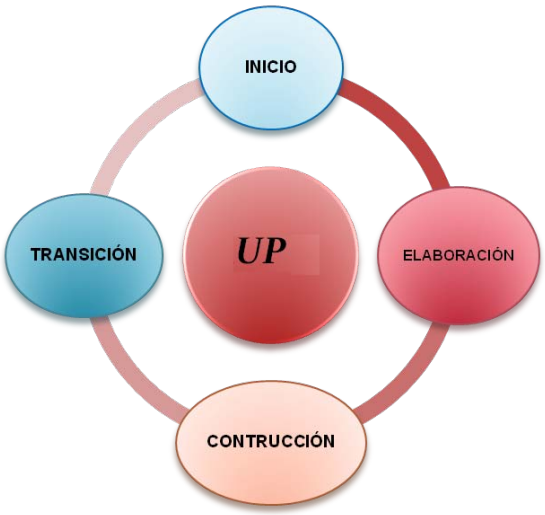

Figura 2.1: Metodología UP.

El desarrollo del proyecto se basó en la metodología UP (ver imagen 2.1), dividiendo los tiempos de desarrollo del proyecto en partes similares.
A continuación se describen las etapas de la metodología UP aplicadas al proyecto:

- **Inicio:** En esta fase analizamos los requerimientos de la aplicación "Pizarrón colaborativo" poniendo énfasis a la característica de plasticidad requerida en la barra de herramientas; además, realizamos un análisis de las posibles soluciones tecnológicas que pudieran aplicarse para el desarrollo de la aplicación.

- **Elaboración:** En esta fase se realizó un primer diseño basado en el análisis de requerimientos. El diseño fue expresado con diagramas de caso de uso y con los diagramas de secuencia correspondiente a cada caso de uso, los cuales sirvieron de base para obtener la primera versión del algoritmo encargado de proporcionar la característica de plasticidad a la barra de herramienta de la aplicación.

- **Construcción:** En esta fase se realizó la programación de la soluión para PC, la cual sirvió de base para implementar en una segunda iteración la programación para la versión de dispositivo móvil.

- **Transición:** En esta fase se revisó que la aplicación se encontrara terminada de acuerdo a lo planteado en los requerimientos en la fase de inicio.

2.2. Análisis

Teniendo en cuenta la heterogeneidad de los dispositivos de cómputo (PC y dispositivos móviles) los cuales cuentan con diferentes capacidades de procesamiento, tamaño, resolución de pantalla, sistema operativo. Para la realización del GUI dinámica se consideró principalmente las características de la pantalla de los dispositivos que a continuación se describen.

- **Resolución de pantalla:** Es el número de pixeles que puede ser mostrado en la pantalla.
- **Tamaño absoluto:** Se refiere al ancho y alto de la pantalla, expresado generalmente en pulgadas.
- **Resolución o tamaño relativo:** Se refiere al número de pixeles que se muestran en la ventana de la pantalla.
- **Tamaño de la ventana:** Se refiere al tamaño de la ventana de la aplicación. En la cual el usuario puede aumentar o disminuir su tamaño.
- **Orientación de la ventana:** Se refiere a las pantallas que pueden cambiar su orientación vertical u horizontal.
- **Versión del sistema operativo:** Se refiere al sistema operativo instalado en los dispositivos.

A partir de las características listadas anteriormente se determinó el tamaño de los interactores[1] y la ubicación de la barra de herramientas

- El tamaño máximo de la ventana se estableció en relación al tamaño relativo de la pantalla.
- El tamaño de los íconos se estableció en relación al tamaño relativo de la pantalla.
- De acuerdo al tamaño de la ventana se agrupan o desagrupan los íconos de la barra de herramientas.
- Con la orientación de la ventana se agrupan o desagrupan los íconos de barra de herramientas.

En general la elección del tamaño de los interactores de la barra de herramientas se determinó de acuerdo con el tamaño relativo de la pantalla de los dispositivos, esto es porque cada dispositivo cuenta con diferente tecnología la cual permite mostrar un mayor o menor numero de pixeles por pulgada; por lo tanto, el tamaño de los interactores expresado en pixeles debe ser mayor en resoluciones grandes y menor en resoluciones pequeñas para así obtener una mejor definición en los interactores.

[1] Un interactor se define como una entidad capaz de mostrar (parte de) su estado, en algún medio de presentación [3]

Resoluciones frecuentemente usadas.

Debido a la gran variedad de pantallas que existe en el mercado se realizó una investigación de las resoluciónes más usadas para determinar el tamaño de los interactores de acuerdo a la resolución de la pantalla.

Resoluciones más usadas en PC.

La versión del algoritmo para PC, considera resoluciones más usadas en el mercado; éstas se eligieron de acuerdo al estudio reportado en la página *w3schools.com* en el mes de enero del 2013 como se observa en la figura 2.2; las resoluciones más usadas en el mercado son 1366X768, 1920X1080 y 1280X1024. Para más detalle ver anexo A el cual muestra parte de este estudio.

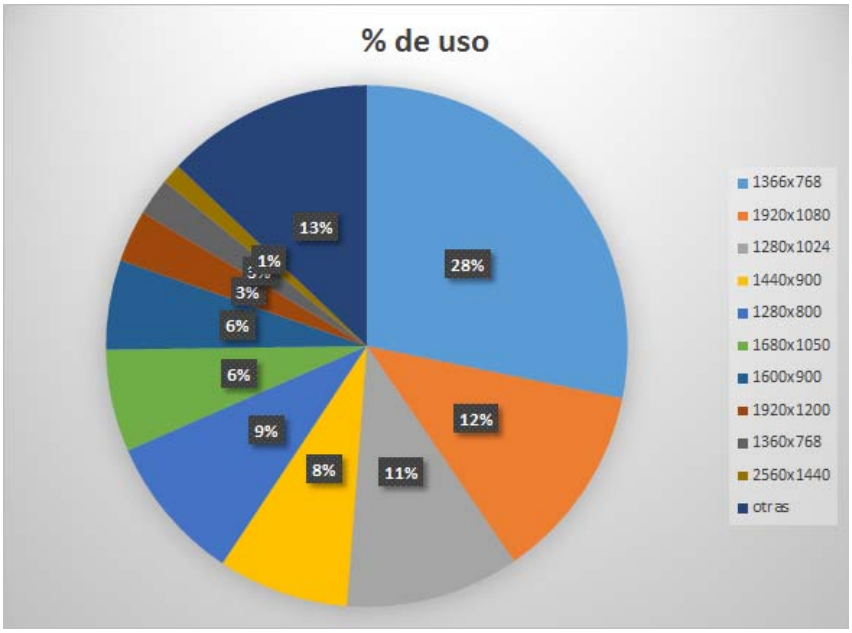

Figura 2.2: Resoluciones más usadas enPC.

En el proyecto solo se realizaron pruebas con 8 resoluciones las cuales se clasificaron como muestra la tabla 2.1 para determinar el tamaño de los íconos a utilizar en cada caso, sin embargo el algoritmo de plasticidad contempla la posibilidad de agregar nuevas resoluciones, solo es necesario crear íconos acorde a la nueva resolución.

Clasificación	Resoluciones	T ícono
Alta	1920x1080	40X40
Media	1366x768	32X32
Media	1440x900	32X32
Media	1280x1024	32X32
Media	1280x800	32X32
Baja	800 x 600	16X16

Cuadro 2.1: Resoluciones PC

Las clasificaciones alta, media y baja fueron determinadas por la correspondencia del tamaño de un ícono para una resolución. Se realizaron pruebas con los tamaños de ícono en todas y cada una de las resoluciones; el ícono que mejor se observó en cada resolución determinaron la clasificación.

Resoluciones más usadas dispositivos móviles.

La versión del algoritmo para móviles considera las resoluciones más usadas en el mercado. Estas se eligieron de acuerdo a un estudio reportado en la página *developer.android.com*, la figura 2.3 muestra que el 80 % de las personas emplean un dispositivo móvil con una resolución Normal (470dpx320dp); y el solo el 9 % emplea una resolución Small (426dpx320dp). Para más detalle ver anexo A, el cual muestra parte de este estudio.

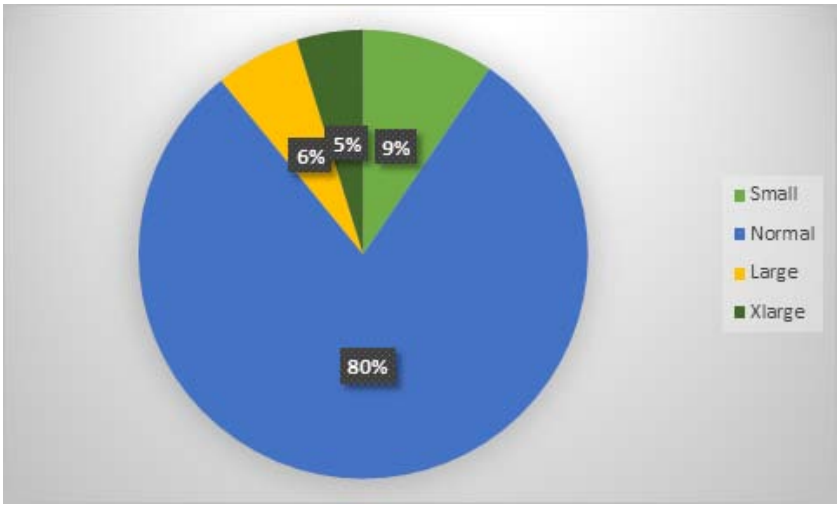

Figura 2.3: Resoluciones más usadas en Moviles

Para la realización del proyecto se utilizó la clasificación de las resoluciones reportada en la página *http://developer.android.com* que se muestra en la tabla 2.2 con la cual se definió el tamaño de ícono a utilizar en cada resolución.

clasificación	Resolucion	T ícono
xlarge	960dp x 720dp	96X96
large	640dp x 480dp	72X72
normal	470dp x 320dp	48X48
small	426dp x 320dp	48X48

Cuadro 2.2: Resoluciones móviles

La decisión de tomar la clasificación fue porque el desarrollo fue para dispositivos Android por tanto el tamaño de los íconos fue de acuerdo con lo reportado en su página oficial.

2.3. Diseño

Diagrama de casos de uso.

A continuación se describe los principales casos de usos de la aplicación.

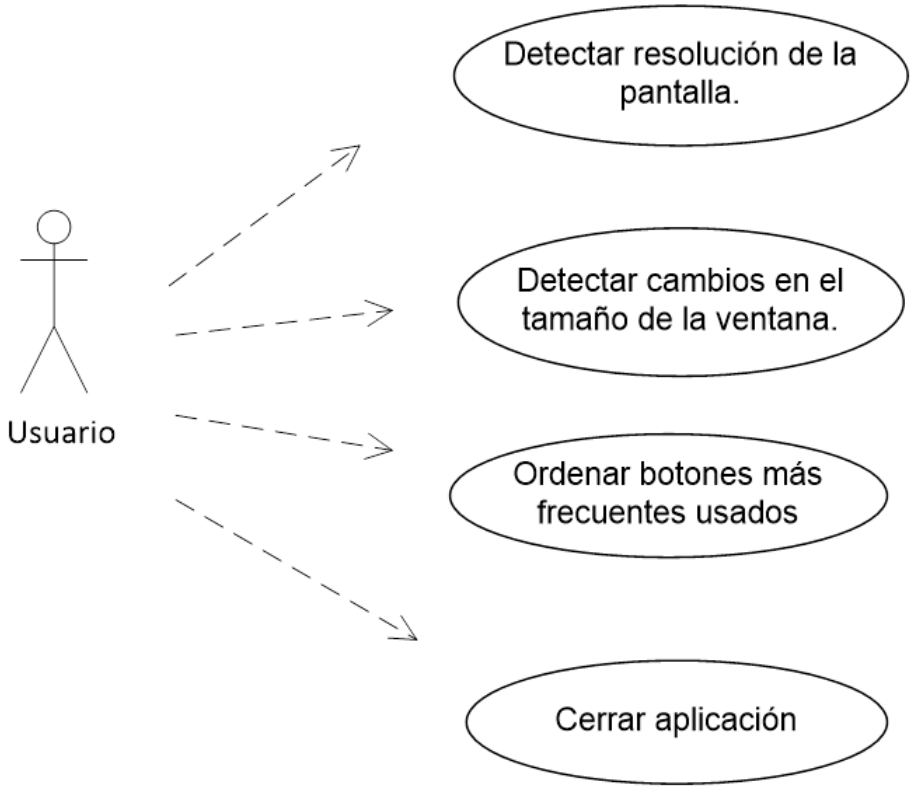

Figura 2.4: Digrama casos usos.

Caso de uso: Detectar resolución de la pantalla.
El usuario inicia la aplicación, ésta detecta la resolución de la pantalla con la cuál calcula el tamaño de los interactores y luego muestra la barra de herramientas con un tamaño n acorde a la resolución del dispositivo.

Caso de uso: Detectar cambios en el tamaño de la ventana.
El usuario cambia el tamaño de la ventana de aplicación; ésta detecta que hay un cambio, verifica el tamaño actual de la ventana y decide agrupar o desagrupar los íconos, además calcula cual es la mejor organización de acuerdo al área de despliegue que se tiene.

Caso de uso: Ordenar íconos siguiendo el criterio del mas usado.
El usuario inicia la aplicación, ésta verifica cual fue la frecuencia de uso de cada botón en la sesión anterior y ordena utilizando el algoritmo de burbuja mandando los íconos más usados al inicio de la barra de herramientas.

Caso de uso: Cerrar aplicación
El usuario cierra la aplicación y ésta termina su ejecución.

Diagrama de Secuencia

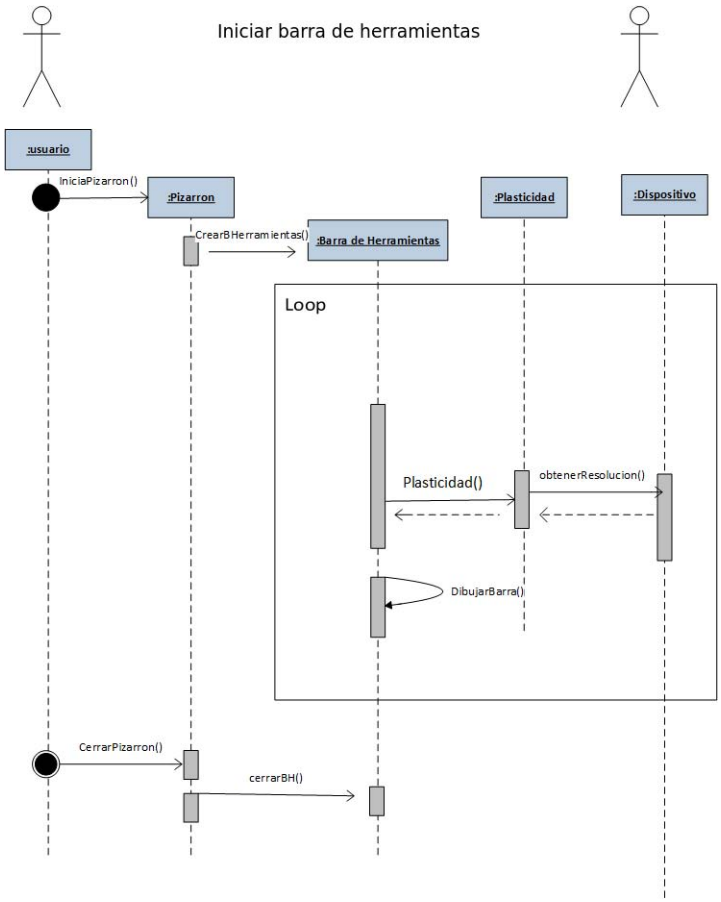

Figura 2.5: Digrama de secuencia

Foma	Elemento	Descripción
1	Usuario	Es la persona que interactúa con la aplicación
2	Pizarrón	Es el objeto principal de la aplicación que manda a llamar al método crear barra
3	Barra de herramienta	Es un objeto de la clase barra de herramientas que se encarga de dibujar y controlar el funcionamiento de la barra de herramientas.
4	Plasticidad	Es un objeto de la clase Plasticidad que se encarga de interactuar con la barra de herramientas para determinar cómo será el despliegue de la misma.
5	Dispositivo	Es el dispositivo (PC o dispositivo móvil) con el que está interactuando la aplicación.

Diagrama de Clases.

Las clases principales de la barra de herramientas en su versión para PC son las siguientes:

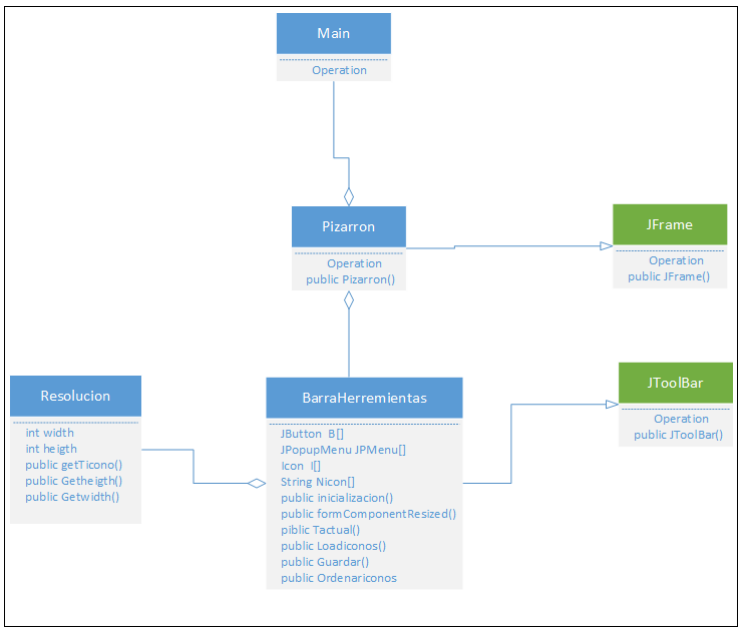

Figura 2.6: Digrama de clases

Pizarrón: Esta clase simula un pizarrón compartido, el cual contiene la barra herramientas. Esta clase hereda sus métodos y atributos de la clase JFrame.

Métodos:

- *Pizarron:* Crea los interactores del pizarrón compartido.

BarraHerramientas: Esta clase contiene el comportamiento de la barra de herramientas. Esta clase hereda sus métodos y atributos de la clase *JBarTool*.

Métodos:

- *inicilizacion:* Crea los interactores que utiliza la barra de herramientas.
- *formComponentResized:* Agrupa o desagrupa los íconos de la barra de herramientas de acuerdo al tamaño actual de la ventana.
- *Tactual:* Obtiene el tamaño actual de la ventana y calcula cual es la agrupación más adecuada de los íconos.
- *Loadiconos:* Carga los nombre de los íconos de la barra de herramientas de un archivo de texto.
- *Guardar:* Guarda los nombres de los íconos cuando se termina la sesión en un archivo de texto.
- *Ordenaiconos:* Ordena los íconos siguiendo el criterio de frecuencia de uso.

Resolución: Esta clase calcula la resolución de pantalla que es utilizada por la clase barra herramientas.

Métodos:

- *getTicono:* obtiene el tamaño de ícono a usar en la barra de herramientas.
- *Getheigth:* obtiene el largo de la pantalla expresado en pixeles.
- *Getwidth:* obtiene el ancho de la pantalla expresado en pixeles.

Las clases principales dela barra de herramientas en su versión para móviles son las siguientes:

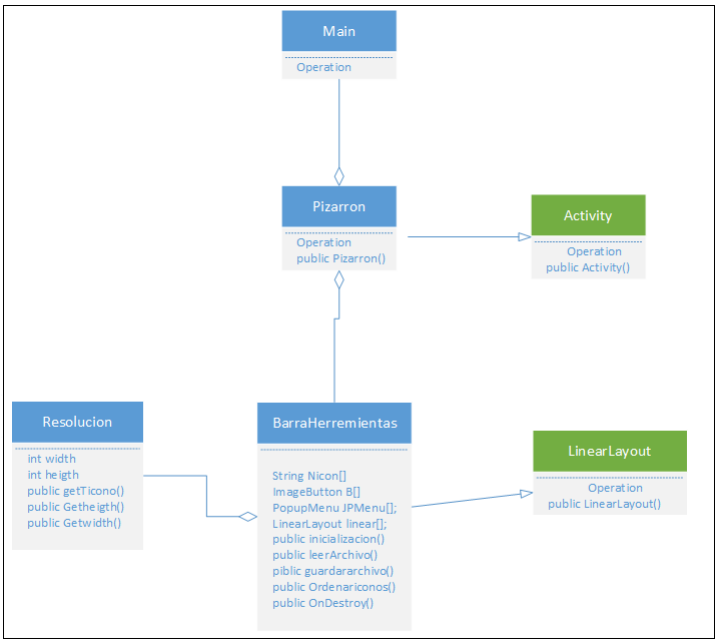

Figura 2.7: Digrama de clases

Pizarrón: Esta clase simula un pizarrón compartido, el cual contiene la barra herramientas. Esta clase hereda sus métodos y atributos de la clase Activity.

Métodos:

- *Pizarron:* Crea los interactores del pizarrón compartido.

Barra Herramientas: Esta clase contiene el comportamiento de la barra de herramientas. Esta clase hereda sus métodos y atributos de la clase LinearLayout.

Métodos:

- *inicilizacion:* Crea los interactores que utiliza la barra de herramientas.
- *leerArchivo:* Carga los nombre de los íconos de la barra de herramientas de un archivo de texto.
- *guardararchivo:* Guarda los nombres de los íconos cuando se termina la sesión en un archivo de texto.
- *Ordenaiconos:* Ordena los íconos siguiendo el criterio de frecuencia de uso.
- *onDestroy:* Llama el metodo *guardarArchivo* y termina la aplicacion.

Resolución: Esta clase calcula la resolución de pantalla que es utilizada por la clase barra herramientas.

Métodos:

- *getTicono:* obtiene el tamaño de icono a usar en la barra de herramientas.
- *Getheigth:* obtiene el largo de la pantalla expresado en pixeles.
- *Getwidth:* obtiene el ancho de la pantalla expresado en pixeles.

2.4. Descripción general de los algoritmos.

Se diseñaron dos algoritmos: Plasticidad barra herramientas y Más frecuente usado .

El algoritmo plasticidad de la barra de herramientas permite desplegar la barra de herramientas dinámicamente de acuerdo al tamaño de la resolución de la pantalla y tamaño de la ventana.

El algoritmo de más frecuente usado permite ordenar los íconos de la barra de herramientas de acuerdo a la frecuencia de uso de la sesión anterior.

Algoritmo de Plasticidad.

Algorithm 1 Algotimo de plasticidad barra de herramientas

```
   getResolucion();
2: Huax = Haux
   Waux= Waux
4: if Waux = Alta then
     Tícono = alto
6: else if Waux = media then
     Tícono = medio
8: else if Waux = baja then
     Tícono = bajo
10: end if
   repeat
12:   if  Waux != Tact then
        if  Tact = 1  then
14:       Agrupación 1 ;
        else if  Tact = 2 then
16:       Agrupación 2;
        else if  Tact= 3 then
18:       Agrupación 3;
          .
20:       .
          .
22:     else if  Tact =n then
          Agrupación n;
24:     end if
      end if
26: until Cerrar = False
```

La descripción del algoritmo se divide en 3 partes inicialización, elección de plasticidad y fin.
Inicialización:

- El algoritmo obtiene la resolución de la pantalla del dispositivo con el método *getResolucion* y le asigna los valores `Haux` y `Wuax` (largo y ancho de la panta).
- Con el valor obtenido en `Waux` pregunta si `Waux` pertenece a la resolución baja, media o alta, y en base lo obtenido se elige el tamaño de los íconos y se asigna el valor a `Tico`.
- Se establece el tamaño de los íconos que tendrá la barra de herramientas.

Elección de plasticidad:

- El algoritmo inicia un ciclo el cual identifica si `Tactual` (tamaño actual de la ventana) cambia o si la aplicación es cerrada.
- Si `Waux` es diferente a `Tactual`, significa que el tamaño de la ventana ha cambiado por lo tanto el algoritmo pregunta `Tactual` está en la clasificación `1, 2, 3 ... n-1` y elige la agrupación `n` de acuerdo a `Tactual.`

Fin: El algoritmo identifica que la aplicación ha sido cerrada por el usuario por el usuario.

- Llama al método `ordenaiconos` el cual ordena los íconos por frecuencia de uso.
- Termina la aplicación.

Algoritmo más frecuente usado.

El algoritmo ordena los íconos de la barra de herramientas por categorias, a saber: herramientas, formas, lineas y colores; el criterio de orden está basado en la frecuencia de uso de cada ícono en la sesión actual. El ordenamiento se realiza a través del algoritmo de burbuja (ver anexo B); además, se guardan los nombres de los íconos en un archivo de texto.

Algorithm 2 Algoritmo mas Frecuente usado

```
    for i = 1 → n do
      for i = 1 → n do
3:      if Fuso(i) ¿Fuso(i+1)  then
          aux = Fuso(i)
          Fuso(j) = Fuso(j+1)
6:        Fuso(j-i) = aux
        end if
      end for
9: end for
```

Observe que que el algoritmo de burbuja se aplica sobre una lista lamada `Fuso` de `n` valores.

Capítulo 3

Implementación

3.1. Alternativas Tecnológicas.

En el proceso de decidir cuál sería la tecnología que cumpliera ampliamente con los objetivos del proyecto, además que redujera tiempos en el desarrollo se consideraron las siguientes alternativas.

Alternativas para dispositivos móviles.

La primer alternativa tecnológica fue MySaiFu, una máquina virtual java para Windows Mobile. Es un software libre bajo la licencia pública de GNU 2 GPLv2. El objetivo principal del proyecto MySaiFu es crear una Máquina Virtual Java para dispositivos con Windows Mobile y Windows CE.

Ventajas:

Nos permite transformar el código .jar de una aplicación hecha en JSE, esto ahorra tiempo de programación y el tener que aprender otro lenguaje en este caso J2ME.

Desventajas:

La principal desventaja y la causa de no realizar la implementación de la barra de herramientas en MySaiFu es que es una tecnología exclusiva de dispositivos móviles con Windows Mobile. El objetivo del proyecto es tener una aplicación multiplataforma; es decir, que se pueda instalar en diferentes dispositivos de cómputo con sistemas operativos diferentes.

3.2. Tecnología empleada: software y hardware.

Software:

La tecnología de Software empleada para el desarrollo de la aplicación fue la siguiente: Para la realización de del código del algoritmo para PC se empleó el lenguaje de programación java, ya que nos ofrece un conjunto básico de herramientas usadas para desarrollar Java Applets, así como las APIs orientadas a la programación de aplicaciones de usuario final: Interfaz gráfica de

usuario, multimedia, redes de comunicación, etc. Con las cuales se cubrieron los requerimientos de implementación para PCs.

Hardware:
La tecnología de Hardware empleada para el desarrollo de la aplicación es la siguiente:
Sattelite A105 :

Lap-Top
Sistema operativos: Windows 8 y Ubuntu 13.04
Procesador: AMD 64 Atlon X2
Memoria RAM : 3 GB
Disco Duro: 250GB

Galaxy TAB :
Sistema Operativo: Android 2.2
Procesador de aplicación de 1.06 Ghz con PowerVR SGX40
Pantalla 7.0 pulgadas WSVGA (1024px X 600px), TOUCH.

3.3. Principales módulos implementados

En esta sección se presenta la descripción del diagrama de paquetes desarrollado para PC y dispositivo móvil, además de ejemplos de módulos desarrollados en lenguaje Java para PC y módulos desarrollados en Android para dispositivos móviles.

3.4. Diagrama de paquetes de PC

A continuación la figura 3.1 muestra el diagrama de paquetes y relaciones entre ellos correspondiente a la versión desarrollada para PC.

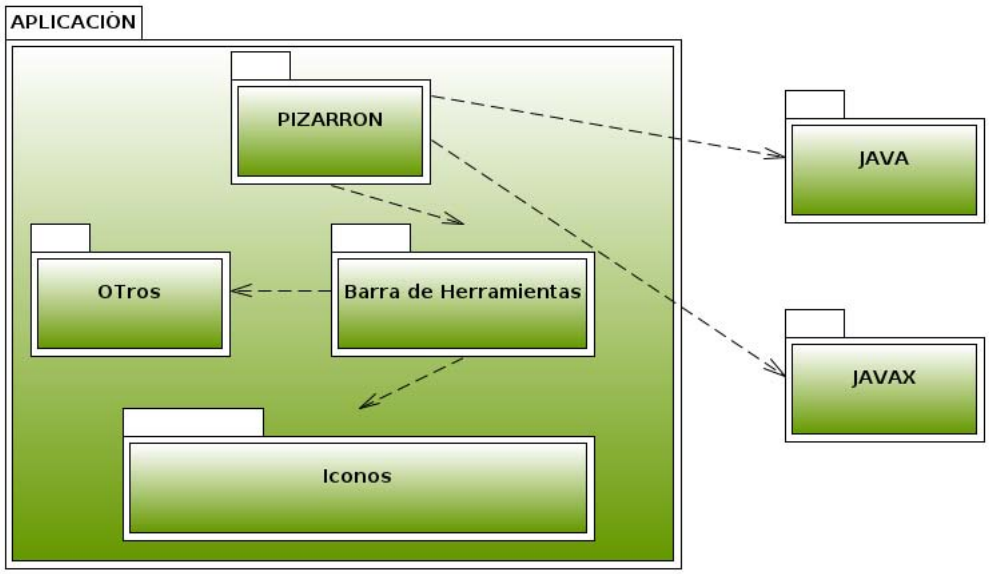

Figura 3.1: Digrama de paquetes PC

- **Aplicación.** Es el paquete de mayor jerarquía. Contiene a todos los paquetes existentes y a la clase Main.
- **PizarronCompartido.** Es el paquete contiene a la clase demo que simula a la aplicación pizarrón compartido
- **Herramientas.** Es el paquete que contiene a la clase `BarraHerramientas`.
- **Otros.** Es el paquete que contiene la clase Resolución.
- **Iconos.** Este paquete contiene los íconos a usar en la aplicación.
- **javax.** Es el paquete externo al cual se recurrió para hacer uso de las clases de la biblioteca swing.
- **java.** Es el paquete externo al cual se recurrió para hacer uso de las clases de la biblioteca awt.

3.5. Ejemplos de módulos programados para PC

Pizarrón compartido: La figura 3.2 muestra fragmentos de código de la clase `Pizarron` y el método *Pizarron* el cual agrega la barra de herramientas como parte del pizarrón.

```
import java.awt.event.*;
import Aplicacion.Otros.Resolucion;
public class Pizarron extends JFrame {
    static BarraHerr BH = new BarraHerr();
    static Resolucion R = new Resolucion();
 public Pizarron(){
    setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    Dimension DMin =new Dimension((int)(R.getwidth()),(int)(R.getheigth()));
    DMin.height=((R.getTiconoB()+15)*7)+R.getTicono()+10;
    DMin.width= (R.getTicono()+15)*4;
    this.setSize(screenSize);
    this.setMinimumSize(DMin);
   // AGREGANDO BARRA DE HERRAMIENTAS
          this.add(BH,BorderLayout.NORTH);
```

Figura 3.2: Clase pizarrón PC

Clase Resolución: La figura 3.3 muestra fragmentos de código de la clase `Resolucion` y sus métodos `getTicono y getTiconoB` encargados de la elección del tamaño de los íconos a usar, de acuerdo a la resolución de la PC. Por ejemplo el método `getTicono` muestra que si el ancho de la pantalla es mayor o igual 1600 pixeles los íconos que se usarán miden 40X40 pixeles.

```
public class Resolucion {
        private int height;
        private int width;

  public int getTicono(){
       Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
        if(screenSize.width >=1600)
              return 40;
        else if (screenSize.width>=1152)
                return 32;
        else if (screenSize.width <=1024)
                return 16;
        else
        return 40;
        }

public int getTiconoB(){
    Dimension screenSize = Toolkit.getDefaultToolkit().getScreenSize();
    if(screenSize.width >=1600)
          return 32;
     else if (screenSize.width>=1152)
         return 16;
    else if (screenSize.width <=1024)
         return 10;
    else
          return 32;
```

Figura 3.3: Clase resolución PC

Clase BarraHerramientas: La figura 3.4 muestra fragmentos de código del método *formComponentResized()* perteneciente a la clase `BarraHerramientas` el cual escucha los cambios que se realizan en el tamaño de la ventana.

```
private void formComponentResized(java.awt.event.ComponentEvent evt) {
         TamPant= Tactual(); // Tamaño actual de la pantalla


switch(TamPant){
        case 1:                                          //caso con el minimo de la pantalla, 4 grupos ocultos
                for ( int i = 0; i <4 ; i++ )
                      B[i].setVisible(true);
                for ( int i = 4; i <29 ; i++ )
                      B[i].setVisible(false);
                     break;
     case 2:                                          //caso colores mostrados 3 grupos ocultos
               for ( int i = 22; i <29 ; i++ )
                     B[i].setVisible(true);
               for ( int i = 4; i <22 ; i++ )
                     B[i].setVisible(false);
                     B[0].setVisible(true);
                     B[1].setVisible(true);
                     B[2].setVisible(true);
                     B[3].setVisible(false);
                     break;
     case 3:                          //caso colores y tamaños mostrados, 2 grupos ocultos
              for ( int i = 19; i <29 ; i++ )
                     B[i].setVisible(true);
              for ( int i = 4; i <19 ; i++ )
                     B[i].setVisible(false);
                     B[0].setVisible(true);
                     B[1].setVisible(true);
                     B[2].setVisible(false);
                     B[3].setVisible(false);
                     break;
     case 4:                                  //caso colores ,tamaños mostrados y formas, 1 grupo oculto
            for ( int i = 15; i <29 ; i++ )
                B[i].setVisible(true);
           for ( int i = 4; i <15 ; i++ )
                 B[i].setVisible(false);
                 B[0].setVisible(true);
                 B[1].setVisible(false);
                 B[2].setVisible(false);
                 B[3].setVisible(false);
                 break;
```

Figura 3.4: Método formComponentResized

Observamos que al iniciar el método `formComponentResized` llama al método `Tactual` el cual devuelve la clasificación del tamaño actual de la venta y lo asigna a `TamPant` y pregunta a que caso pertenece por ejemplo si `TamPant = 1` oculta los íconos `B[i]` mediante un bucle que va de `i =4 a i <29` y muestra los íconos de `i = 0 a i <4`, en este caso la ventana solo cuenta con el espacio para mostrar a los íconos agrupados.

3.6. Diagrama de paquetes para Móviles

A continuación la figura 3.5 muestra el diagrama de paquetes correspondiente a la versión desarrollada para móviles.

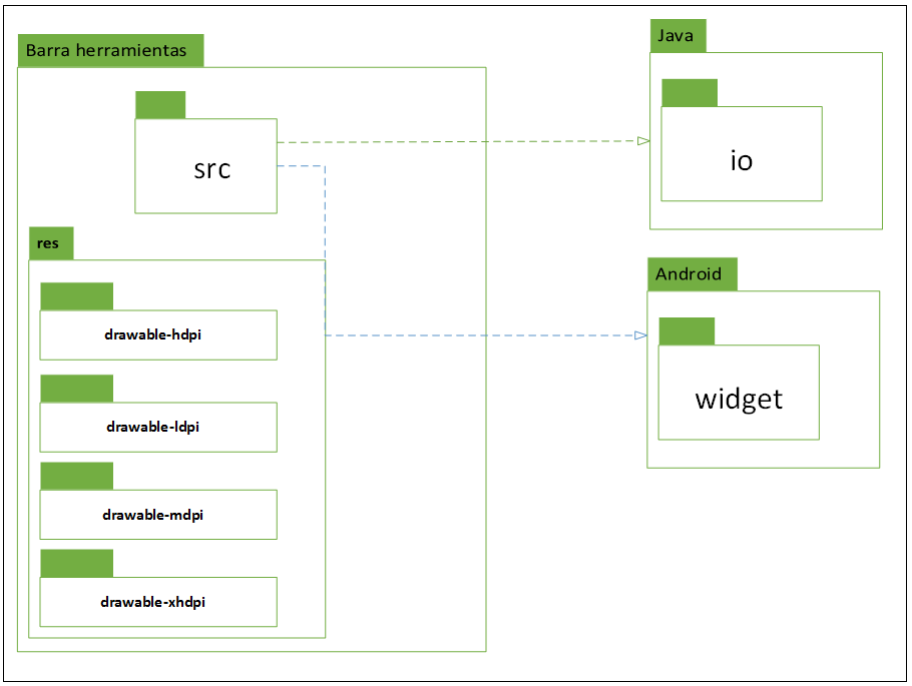

Figura 3.5: Digrama de paquetes PC

Barra herramientas: Es el paquete de mayor jerarquía. Contiene a los paquetes src y res.

Src: Contiene las clases `MainActivity` y `plasticidad` las cuales se encargan de crear la barra de herramientas.

Res: Contiene los recursos que se utilizan en la creación de la barra de herramientas.

Android. Es el paquete externo que proporciona algunas clases de la biblioteca widget, como son: `ScrollView,Button y LinearLayout`

Java. Es el paquete externo que proporciona algunas clases de la biblioteca io, como son: `FileInputStream y FileOutputStream`

3.7. Ejemplos de módulos programados para Móviles

Pizarrón compartido: La figura 3.6 muestra el código de la clase `MainActivity`, la cual simula un pizarrón compartido.

```
import android.widget.TableRow;
import android.widget.Toast;

public class MainActivity extends Activity {
    static final int READ_BLOCK_SIZE = 100;
    String Nicon[]={ "flecha","lapiz","goma","letra","bote"
            ,"tijeras","zoom","lupa","capas","hacer"
            ,"rehacer" // 4-14 Herramientas
            ,"linea","cuadrado","triangulo","circulo" // 15-18
            ,"linea1","linea2","linea3"  // 19-21 Formas
            ,"negro","rojo","verde","gris","azul"
            ,"verdeol","cafe"};
            int C[]={0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0};

            ImageButton B[];
            PopupMenu JPMenu[];
            LinearLayout linear[];

            @Override
            protected void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.activity_main);

                leerArchivo();
                inicilizacion();
```

Figura 3.6: Clase MainActivity

En el código podemos observar el método *onCreate*, el cual ejecuta los métodos *leerArchivo* que se encarga de recuperar los nombre de los íconos. Tambien podemos observar el método *inicialización* que se encarga de crear la barra de herramientas.

Método ordena íconos: La figura 3.7 muestra el código del método *ordenaiconos* el cual se encarga de ordenar los íconos cuando la aplicación es cerrada por el usuario.

```
//////////////////////////////////Ordena bononesS  por frecuencia de uso.
public void ordenaiconos(int inicio, int Tbloque){
String TEMP;
int temp;

        for (int  i =inicio; i < Tbloque; i++) {
                for (int j = i+1; j <Tbloque; j++) {
                    if(C[j] > C[i]){
                        temp = C[j];
                        C[j] = C[i];
                        C[i]=  temp;
                        TEMP= Nicon[j];
                        Nicon[j] = Nicon[i];
                        Nicon[i]=  TEMP;
                        }//fin if
                    }// fin 2 for
        }//fin 1 for
}
```

Figura 3.7: Método ordena íconos

Nuevamente se empleó el algoritmo de burbuja para realizar el ordenamiento. Se tienen dos arreglos `Nicon` y `C`. `Nicon` guarda los nombre de los íconos, y `C` guarda la frecuencia de uso de cada nombre de ícono. El algoritmo compara cada elemento del arreglo `C[i]` con `C[j]`, si el primero es menor que el segundo entonces sus valores son intercambiados; esto implica intercambiar los nombres de ícono correspondientes en el arreglo `Nicon`.

Método inicialización: La figura 3.8 muestra fragmentos del código del método *inicialización* el cual es encargado de crear dinámicamente la barra de herramientas.

```
DisplayMetrics metrics = new DisplayMetrics();
getWindowManager().getDefaultDisplay().getMetrics(metrics);

switch(metrics.densityDpi)
{
case DisplayMetrics.DENSITY_XHIGH:
         Ticono=96;
         break;
case DisplayMetrics.DENSITY_HIGH: //HDPI
    Ticono= 72;
    break;
case DisplayMetrics.DENSITY_MEDIUM: //MDPI
    Ticono= 48;
    break;

case DisplayMetrics.DENSITY_LOW:  //LDPI
    Ticono= 48;
    break;
}
```

Figura 3.8: Método inicialización

Se observa una estructura de control `case`, la cual determina el tamaño de los íconos de acuerdo al tipo de pantalla; por ejemplo, si la pantalla es `DENSITY_XHIGH` el `Ticono` (tamaño de ícono) a usar es de 96x96 pixeles.

Capítulo 4

Resultados y Pruebas

En las siguientes secciones se describe las pruebas realizadas a la barra de colaboradores en su versión para PC y dispositivos móviles.

4.1. Pruebas para Dispositivo PC.

La figura 4.1 muestra la barra de herramientas maximizada por lo cual la aplicación identifica que tiene suficiente espacio y no realiza ninguna agrupación.

Figura 4.1: Ventana maximizada

La figura muestra 4.2 la barra de herramientas en 4 cuadrantes en cada uno de ellos el tamaño de la pantalla es menor por lo cual la aplicación realiza la agrupación.

- **Cuadro I:** La barra de herramientas muestra agrupando los íconos de herramientas.
- **Cuadro I:** La barra de herramientas muestra agrupando los íconos de herramientas y formas.
- **Cuadro III:** La barra de herramientas muestra agrupando los íconos de herramientas, formas y líneas.
- **Cuadro IV:** La barra de herramientas muestra agrupando los íconos de herramientas, formas, líneas y colores.

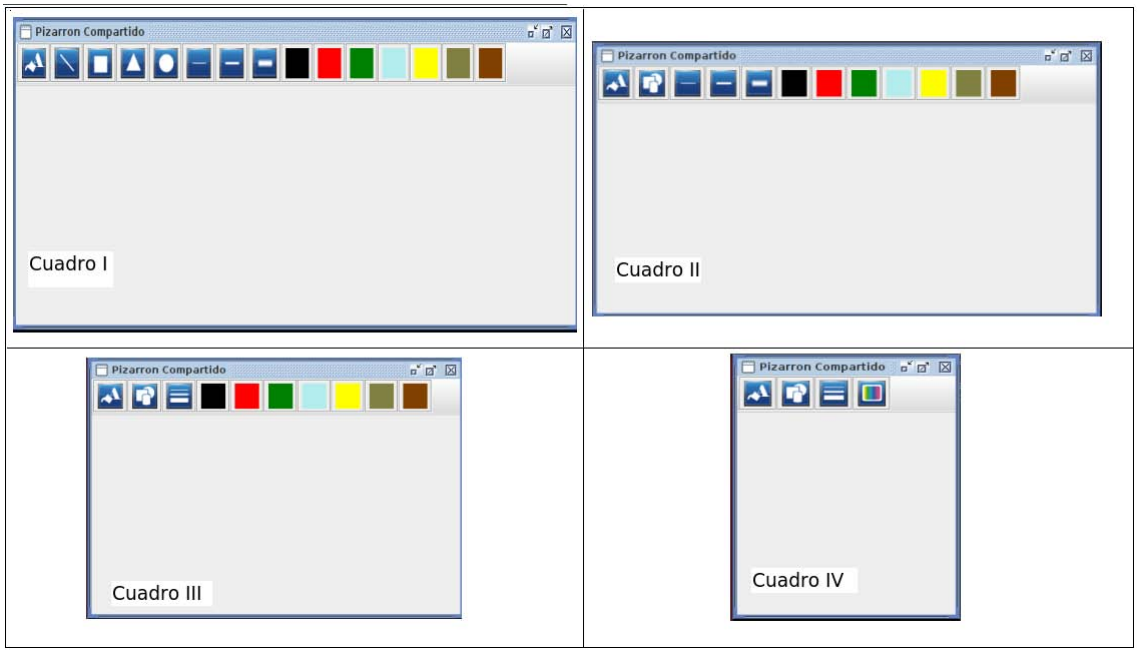

Figura 4.2: Ventana 1

El algoritmo de plasticidad permite que una vez que los íconos se encuentren agrupados, estos se muestren organizados de la mejor manera tomando en cuenta el tamaño actual de la ventana. A continuacion se explica una prueba de los antes descrito.

La figura 4.3 muestra la barra de herramientas cuando ya se realizó la agrupación de los íconos. Se muestran 4 cuadros en cada uno de ellos el tamaño de la ventana es menor, por lo cual la aplicación ordena los íconos del grupo herramientas de acuerdo al tamaño de la ventana.

- **Cuadro I:** Los íconos del grupo herramientas se muestran en línea horizontal.
- **Cuadro II:** Los íconos del grupo herramientas se muestran en línea horizontal colocando 2 íconos en la segunda línea.
- **Cuadro III:** Los íconos del grupo herramientas se muestran en línea horizontal colocando 3 íconos en la segunda línea.

- **Cuadro IV:** Los íconos del grupo herramientas se muestran en línea vertical. colocando 5 íconos en la segunda línea

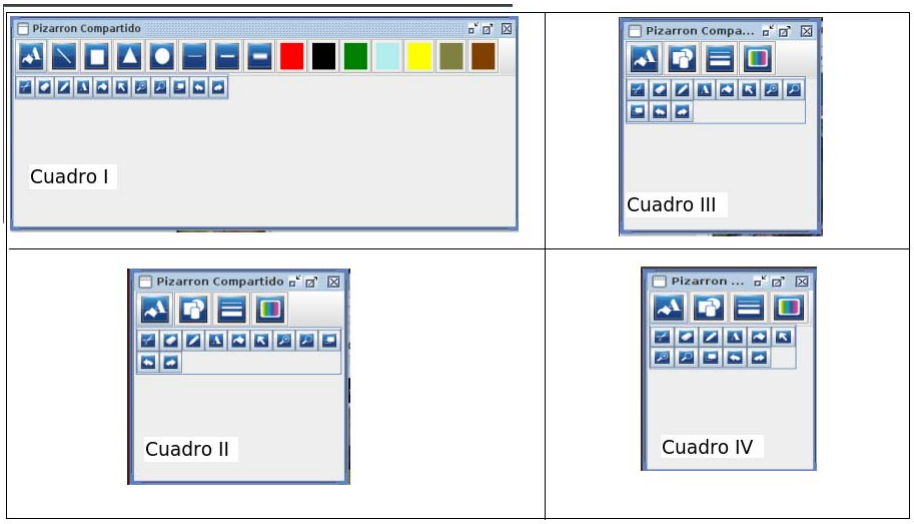

Figura 4.3: Ventana 1

Al iniciar la aplicación se deben mostrar los íconos ordenados por la frecuencia que tuvieron en la sesión anterior. A continuación se explica una prueba de lo antes descrito.

La figura 4.1 muestra la barra de herramientas cuando se ejecutó por primera vez; la figura 4.4 muestra la barra ejecutada una segunda vez, donde los íconos: tijera,
goma y lápiz ocupan las primeras posiciones, ya que fueron los que más usados en la sesión anterior.

Figura 4.4: Ventana 2

4.2. Pruebas para Dispositivo Móvil.

La figura 4.5 muestra la barra de herramientas horizontalmente corriendo en un dispositivo con sistema operativo Android, con una pantalla `DENSITY_XHIGH` por lo tanto los íconos son de un tamaño 96x 96 pixeles.

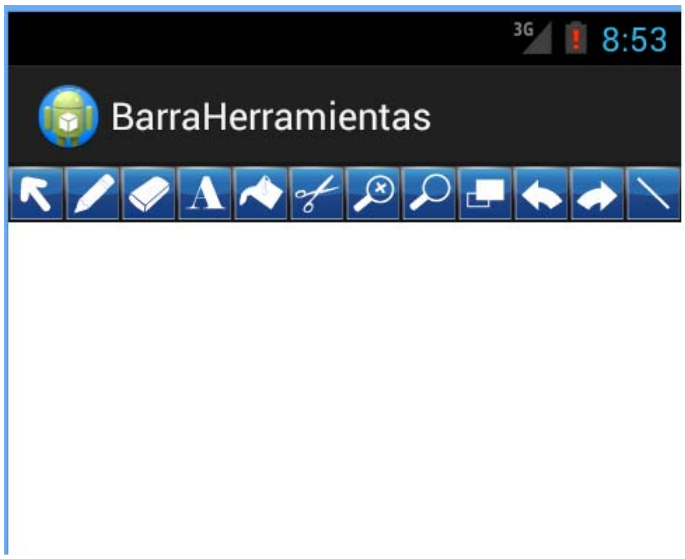

Figura 4.5: Barra de herramientas

En el caso de los dispositivos móviles y debido al espacio que se tiene para mostrar la barra de herramientas se optó por no realizar la agrupación de los íconos, en lugar de eso se dotó de un scroll horizontal como se observa en la figura 4.6.

Figura 4.6: Barra de herramientas

Al iniciar la aplicación se deben mostrar los íconos ordenados por la frecuencia que tuvieron en la sesión anterior. A continuación se explica una prueba de lo antes descrito.

La figura 4.5 muestra la barra de herramientas cuando se ejecutó por primera vez; la figura 4.7 muestra la ejecución de la barra por segunda vez, donde los íconos: tijera,
goma y lápiz ocupan las primeras posiciones, ya que fueron los que más usados en la sesión anterior.

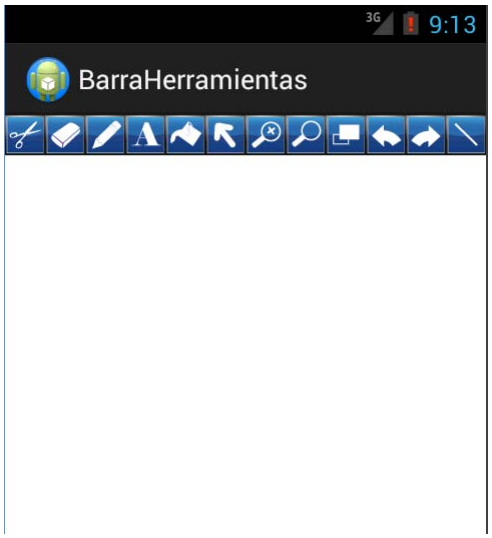

Figura 4.7: Ventana 2

Capítulo 5

Conclusiones y Trabajo Futuro

El presente proyecto tuvo como objetivo general el dotar a la aplicación pizarrón compartido con una barra de herramientas que se despliegue dinámicamente en función del área del dispositivo. A continuación se muestran los objetivos específicos, así como el alcance que se obtuvo con el de desarrollo de este proyecto terminal.

Objetivo: Realizar pruebas de usabilidad sobre tamaños y posiciones de los diferentes interactores que conforman la barra de herramientas.

Descripción: Se estudiaron distintos dispositivos para así determinar las posibilidades de implementar los elementos de información. A partir de estas pruebas se diseñaron los prototipos de GUI para PC y dispositivo móvil.

Status: Completo.

Objetivo: Diseñar e implementar un algoritmo que permita desplegar dinámicamente la barra de herramientas, considerando el tamaño del dispositivo.

Descripción: Se diseñó el algoritmo de acuerdo a los casos de uso presentados. Además, se realizaron pruebas sobre resoluciones y tamaños de pantalla de los dispositivos para mejorar y adaptar el algoritmo de la creación de la barra de herramientas.

Status: Completo.

Objetivo: Realizar pruebas para determinar la ubicación del algoritmo (cliente-servidor versus peer to peer.

Descripción: Las pruebas no fueron realizadas porque éstas debieron realizarse en colaboración con las personas que desarrollaron los diferentes módulos del pizarrón compartido; sin embargo, cada colaborador realizó su módulo en momentos diferentes.
Status: No alcanzado.

Objetivo: Diseñar e implementar una segunda etapa del algoritmo para medir la frecuencia de uso que, sobre los interactores, se realiza en cada sesión el usuario.

Status: Completo.

Objetivo: Migrar el algoritmo a la aplicación de pizarrón compartido.

Descripción: La migración no fue realizada porque esta tarea debió realizarce en colaboración con las personas que desarrollaron los diferentes módulos del pizarrón compartido; sin embargo, la aplicación barra de herramientas se diseñó e implementó de forma modular procurando un bajo acoplamiento entre los modulos; por lo tanto, otra persona podrá integrar la barra de herramientas al pizarrón compartido.

Status: No alcanzado.

Trabajo futuro del proyecto.

- En la barra de herramientas para PC y móvil se colocaron una gran variedad de íconos los cuales actualmente no contienen funcionalidad, se espera que en un posterior trabajo se ponga en operación la funcionalidad de estos.
- Lograr la comunicación de aplicación a través de diversos dispositivos.
- Realizar la integración con todos los componentes de la aplicación pizarrón compartido.
- Guardar en una base de datos el tipo de íconos, tamaño de pantallas, frecuencia uso de interactores, etc.

Glosario

Clase: Definiciones de las propiedades y comportamiento de un tipo de objeto concreto. La instanciación es la lectura de estas definiciones y la creación de un objeto a partir de ellas.

Herencia: Es un mecanismo que permite la definición de una clase a partir de la definición de otra ya existente. La herencia permite compartir automáticamente métodos y datos entre clases, subclases y objetos.

Objeto: Entidad provista de un conjunto de propiedades o atributos (datos) y de comportamiento o funcionalidad (métodos) los mismos que consecuentemente reaccionan a eventos. Se corresponde con los objetos reales del mundo que nos rodea, o a objetos internos del sistema (del programa). Es una instancia a una clase.

Método: Algoritmo asociado a un objeto (o a una clase de objetos), cuya ejecución se desencadena tras la recepción de un mensaje. Desde el punto de vista del comportamiento, es lo que el objeto puede hacer. Un método puede producir un cambio en las propiedades del objeto, o la generación de un evento con un nuevo mensaje para otro objeto del sistema.

Evento: Es un suceso en el sistema (tal como una interacción del usuario con la máquina, o un mensaje enviado por un objeto). El sistema maneja el evento enviando el mensaje adecuado al objeto pertinente.

Propiedad o atributo: Contenedor de un tipo de datos asociados a un objeto (o a una clase de objetos), que hace los datos visibles desde fuera del objeto y esto se define como sus características predeterminadas, y cuyo valor puede ser alterado por la ejecución de algún método.

Un píxel o pixel: (acrónimo del inglés picture element, elemento de imagen).Es la menor unidad homogénea en color que forma parte de una imagen digital, ya sea de una fotografía, un fotograma de vídeo o un gráfico.

Densidad de la pantalla: La cantidad de pixeles en un área física de la pantalla, normalmente se conoce como DPI (puntos por pulgada).

Apéndice A

Estadísticas de resoluciones

Estadísticas de resoluciones, fuente: W3Schools [4] al mes de enero de 2013

Resolution	January 2013
1366x768	25.4 %
1920x1080	11.0 %
1280x1024	9.7 %
1440x900	7.3 %
1280x800	8.2 %
1680x1050	5.7 %
1600x900	5.0 %
1920x1200	2.9 %
1360x768	2.1 %
2560x1440	1.1 %
Other high resolutions	11.6 %
Sum	**90.0 %**

Figura A.1: Estadística de resoluciones PC

Estadísticas de resoluciones, fuente: developer.android.com [5] al mes de octubre de 2013

	ldpi	mdpi	tvdpi	hdpi	xhdpi	xxhdpi	Total
Small	9.4%						9.4%
Normal	0.1%	15.3%		33.5%	22.8%	7.7%	79.4%
Large	0.6%	3.5%	1.2%	0.5%	0.6%		6.4%
Xlarge		4.4%		0.3%	0.1%		4.8%
Total	10.1%	23.2%	1.2%	34.3%	23.5%	7.7%	

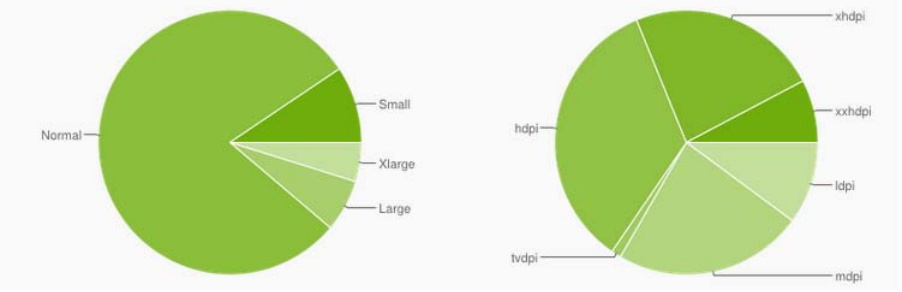

Figura A.2: Estadística de resoluciones móvil

Apéndice B

Ordenamiento de burbuja

La figura B.1 muestra el algoritmo de burbuja, el cual se es utilizado para ordenar los bonotes siguiendo el criterio del más usado, fuente: http://es.wikipedia.org [6].

```
procedimiento DeLaBurbuja2 (a(0), a(1), a(2), ..., a(n−1))
  i ← 1
  ordenado ← no
  mientras (i < n) ∧ (ordenado = no) hacer
    i ← i + 1
    ordenado ← si
    para j ← 0 hasta n − i hacer
      si a(j) > a(j+1) entonces
        ordenado ← no
        aux ← a(j)
        a(j) ← a(j+1)
        a(j+1) ← aux
      fin si
    fin para
  fin mientras
fin procedimiento
```

Figura B.1: Ordenamiento de burbuja

Bibliografía

[1] , G. Calvary, J. Coutaz, O. Daassi, V. Ganneau, L. Balme, A.Demeure and J.-S. Sottet, Métamorphose des IHM et Plasticité: Article de synthése, Ergo'IA 2006, pp. 11-18, 2006.

[2] Gabriela Sánchez Morales (febrero de 2009), Redistribución semi-plásastica mixta: el caso de estudio de un pizarrón compartido (Tesis de Maestría-Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional Unidad Zacatenco) http://www.cs.cinvestav.mx/Estudiantes/TesisGraduados/2009/tesisGabrielaSanchez.pdf. Consultado en el mes de octubre de 2013.

[3] D. J. Duke and M. D. Harrison, 1993 Abstract interaction objects Computer Graphics Forum, Vol 12, No. 3, pp.25-36, 1993.

[4] **Resoluciones PC** http://www.w3schools.com/browsers/browsers_resolution_higher.asp. Consultado en el mes de octubre de 2013.

[5] **Resoluciones móvil** http://developer.android.com/about/dashboards/index.html. Consultado en el mes de octubre de 2013.

[6] **Ordenamiento de burbuja** http://es.wikipedia.org/wiki/Ordenamiento_de_burbuja. Consultado en el mes de octubre de 2013.

[7] D. J. Duke and M. D. Harrison, 1993 **Abstract interaction objects** Computer Graphics Forum, Vol 12, No. 3, pp.25-36, 1993.

[8] **Documentación de Java Standard Edition.** http://java.sun.com/javase. Consultado en el mes de octubre de 2013.

Chapter 1

Manual de instalación

1.1 Instalación del software requerido para la aplicación.

Para la versión PC, se proporciona el archivo PizarronCompartido.jar, para ejecutarlo se necesita tener instalada la JVM (Java Virtual Machine). El procedimiento para instalar la máquina virtual de Java es descrito a continuación:

Linux

Existen 2 formas:

1. Ir a synaptic y buscar el JRE (Java Runtime Enviroment) e instalar el software.
2. Abrir una consola y ejecutar el siguiente comando
sudo apt-get install sun-java6-jre sun-java6-plugin sun-java6-fonts.

Windows

1. Descargar de la página oficial http://www.oracle.com/technetwork/java/index.html el JRE (Java Runtime Enviroment).
2. Ejecutar el archivo descargado.
3. Seguir el procedimiento indicado por el instalador.

1.2 Proceso de instalación de la aplicación.

Instalación de la aplicación barra de herramientas en PC

En Windows :

1. Acceder al directorio "Códigos Fuente/PC" PizarronCompartido.jar.

2. Doble clic sobre el icono de PizarronCompartido (ver figura 1.1).

3. A continuacióon se abrirá la aplicación (ver figura 1.2).

En Linux :

1. Acceder al directorio "Códigos Fuente/PC" barraherramientas.jar ver figura 1.1

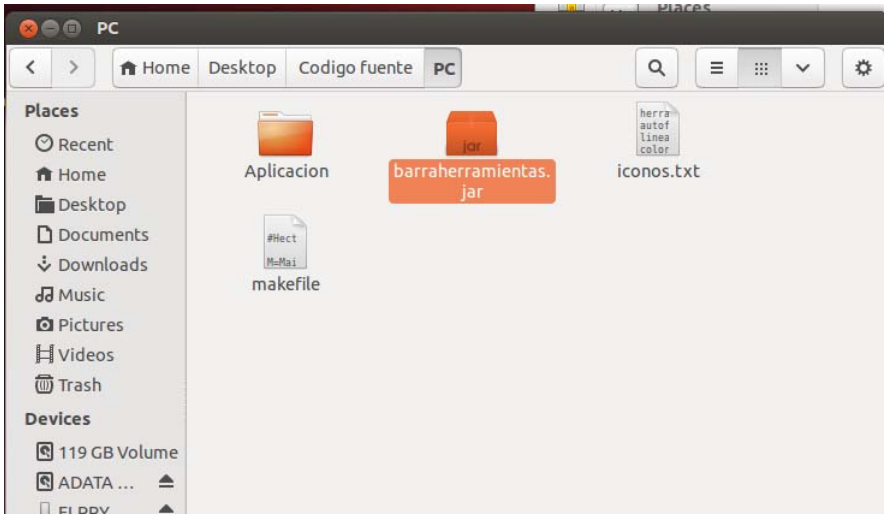

Figure 1.1: Abrir barra de herramientas en Linux

Existen 2 formas de abrir la aplicación

a) Clic sobre el ícono y clic en "Abrir con Open JDK Java 6 Runtime"

b) En una consola acceder a la dirección "Códigos Fuente/PC' y escribir: java -jar PizarronCompartido.jar.

3. A continuacióon se abrirá la aplicación ver figura 1.2

Figure 1.2: Aplicación barra de herramientas en Linux

Instalación de la aplicación en móvil.

- Acceder al directorio "/Codigo fuente/BarraHerramientas/bin.
- Localizar el archivo BarraHerramientas.apk.
- Conectar la unidad de memoria del movil.
- Copiar el archivo BarraHerramientas.apk al dispositivo móvil.
- Desconectar la unidad de memoria del dispositivo móvil.
- Ir al menú de aplicaciones.
- Ir a mis archivos.
- Encontrar el directorio donde fue ubicado el archivo BarraHerramientas.apk
- Clic sobre el archivo a instalar.
- La pantalla mostrará la opción instalar, hacer clic sobre ella.
- A continuación la pantalla mostrará la leyenda Aplicación instalada". Clic en abrir.
- A continuación se abrirá la aplicación de pizarrón compartido.
- La aplicación aparecerá dentro del menú de aplicaciones.

Chapter 2

Manual de usuario

Este manual de usuario explica la forma de usar la barra de colaboradores perteneciente a la aplicación pizarrón compartido en sus dos versiones PC y dispositivo móvil.

PC

La barra de herramientas en su versión PC está dotada principalmente agrupación de los botones como se explica a continuación:

La figura 2.1 muestra la barra de herramientas cuando ya se realizó la agrupación de los botones. Se muestran 4 cuadros en cada uno de ellos el tamaño de la ventana es menor, por lo cual la aplicación ordena los botones del grupo herramientas de acuerdo al tamaño de la ventana.

- **Cuadro I:** Los íconos del grupo herramientas se muestran en línea horizontal.
- **Cuadro II:** Los íconos del grupo herramientas se muestran en línea horizontal colocando 2 botones en la segunda línea.
- **Cuadro III:** Los íconos del grupo herramientas se muestran en línea horizontal colocando 3 botones en la segunda línea.
- **Cuadro IV:** Los íconos del grupo herramientas se muestran en línea vertical. colocando 5 botones en la segunda línea

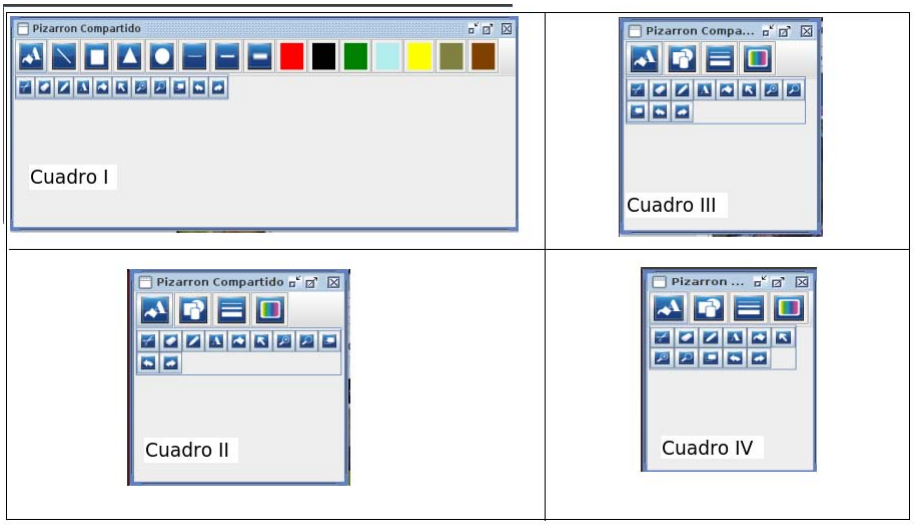

Figure 2.1: Ventana 1

Al iniciar la aplicación se deben mostrar los íconos ordenados por la frecuencia que tuvieron en la sesión anterior. A continuación se explica una prueba de lo antes descrito.

La figura 2.2 muestra la barra de herramientas cuando se ejecutó por primera vez: la figura 2.3 muestra la barra ejecutada una segunda vez, donde se muestran los íconos; tijeras, goma y lápiz en las primeras posiciones ya que fueron los que más se usaron en la sesión anterior.

Figure 2.2: Ventana maximizada

Figure 2.3: Ventana 2

Además cuenta con los botones minimizar, maximizar y cerrar.

Figure 2.4: botones

Móvil

La barra de herramientas en su versión móvil está dotada scroll horizontal como se observa en la figura 2.5.

Figure 2.5: Barra de herramientas

Al iniciar la aplicación se deben mostrar los botones ordenados por la frecuencia que tuvieron en la sesión anterior. A continuación se explica una prueba de lo antes descrito.

La figura 2.5 muestra la barra de herramientas cuando se ejecutó por primera vez; la figura 2.6 muestra la ejecución de la barra por segunda vez, donde los íconos; tijeras, goma y lápiz en las primeras posiciones ya que fueron los que más se usaron en la sesión anterior.

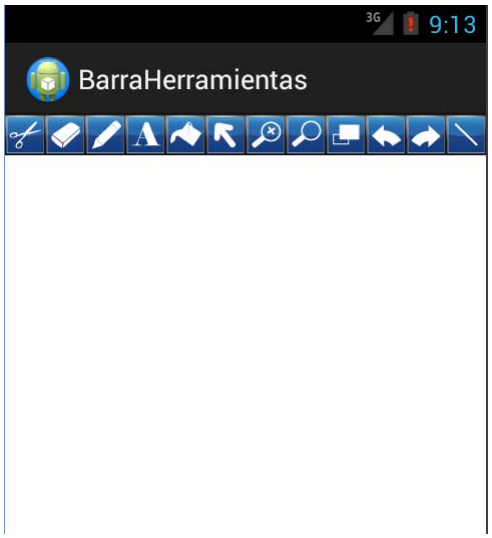

Figure 2.6: Ventana 2