Universidad Austral

Aplicación móvil para manejo de una computadora "Droid Control"
Trabajos de Cátedra

Autores:

- **Tomas A. Najun**
- **Jose M. Gonzalez**

Docentes:

- **Nicolas Damonte**
- **Lucas Luppani**
- **Ignacio Rodriguez**

1. Introducción

A lo largo de los últimos años nos hemos acostumbrado a que nuestros dispositivos móviles cumplan cada vez más funciones. Hoy en día, engloban funciones tan variadas como la telefonía, la cámara de fotos, navegación web y muchos más.

Nuestra aplicación tiene el propósito seguir esta tendencia, al llevar un dispositivo móvil a cumplir la función de distintos periféricos. Lo que se traduce en una mayor comodidad para el usuario y un beneficio económico, gracias a que ya no debe adquirir dichos periféricos.

En síntesis, nuestra aplicación presenta la novedad de incluir una variedad de soluciones periféricas en una sola y sencilla. Permite el manejo a distancia de virtualmente cualquier función de la computadora, así como también le proporciona nuevas funciones a esta.

Este proyecto surge como el trabajo final de una materia (Laboratorio I) de 3º año de Ingeniería Informática. Los objetivos académicos que nos llevaron a implementar esta aplicación fueron:

- La implementación de nuestro conocimiento de programación orientada a objetos.
- Aplicación de conceptos vinculados a la ingeniería de software.
- Confección de un diagrama UML y ser capaces de llegar del alto al bajo nivel.
- Experiencia en el contacto con distintos protocolos de conexión.
- Ímpetu por trabajar con nuevas tecnologías como Android.

2. Propuesta de solución

Nuestra propuesta es la de una aplicación móvil y una de escritorio, que permita al usuario usar la aplicación móvil para controlar distintas funciones de la computadora.

Para el correcto funcionamiento ambos dispositivos deben conectarse haciendo uso del mismo router. Una vez iniciada la aplicación de escritorio, o como la solemos llamar el “Server”, esta hace uso de la conexión WIFI y abre un puerto específico. En el cual se aguarda la comunicación proveniente de la aplicación móvil o “Client”.

Luego de establecer la conexión entre ambas, el usuario puede escoger entre una serie de funcionalidades que le ofrece el programa.

Casos de uso:

- Puede funcionar como mouse y/o teclado a distancia
- Puede pasar a cumplir la función de una Tableta Gráfica Digitalizadora
- Permite manejar un reproductor de medios
- Permite navegar una presentación o hacer alguna señalización en esta
- Puede ser utilizado como control para videojuegos y hasta como volante para juegos de carreras, gracias al acelerómetro del dispositivo móvil.
- El usuario también puede crear sus propios botones en el dispositivo móvil, que le permitan desencadenar los eventos que él decida.

En todo momento el usuario puede terminar la conexión tanto desde el escritorio como desde el dispositivo móvil.

3. Metodología del trabajo

Existen aplicaciones con alguna similitud en el mercado, como ser el “TeamViewer”[7] que está más orientado a la asistencia remota o el “Remote Mouse”[8] que permite el control a distancia del puntero.

Al testear dichas y otras aplicaciones de esta índole nos percatamos de que suelen ser difíciles de manejar, generan muchos errores y el funcionamiento es altamente ineficiente.

Nos propusimos resolver estos problemas, englobar todas las soluciones en una y además agregarle más funciones, para explotar al máximo su potencial.

Nuestra aplicación, permite tanto el control del puntero como de muchas otras funciones. Contrario al "TeamViewer", proporciona una sencilla e intuitiva interfaz de usuario, para el manejo de las funciones de computadora. También, agrega más funcionalidades a la computadora, como ser el uso del acelerómetro.

4. Metodología del trabajo

Para desarrollar una aplicación de esta índole tuvimos que desarrollar, por un lado, una aplicación para la computadora y por otro una para el dispositivo móvil.

En cuanto al código de computadora decidimos programar en Java, tomando provecho así del "Compile ones and run everywhere", de modo que, nuestro programa puede correr en cualquier sistema operativo con una máquina virtual de Java. De esta manera redujimos la complejidad para el usuario y el problema de tener que llevar el mantenimiento para distintas versiones para cada sistema operativo, entre otros problemas.

La aplicación para móviles está programada en Android, de momento, aunque están en desarrollo las versiones para otros sistemas como IOS.
Para programar en Android, se debe definir en primera instancia el nivel de API (Aplication Programing Interfaze) que soportara la aplicación, esta podrá correr también en todas las versiones posteriores no las anteriores[1].

Para nuestra aplicación hemos decidido utilizar la versión 2.3 (ginger bread). Esta versión es la más difundida y actualmente es utilizada por casi un 80% de los dispositivos con Android[1]. Otras razones para que sea la mínima versión solicitada son, que su máquina virtual presenta una mejora sustancial en el recolector de basura, haciendo que las aplicaciones tengan una mejor performance, y que su funcionamiento en tablets y dispositivos de mayor tamaño este optimizado[1].

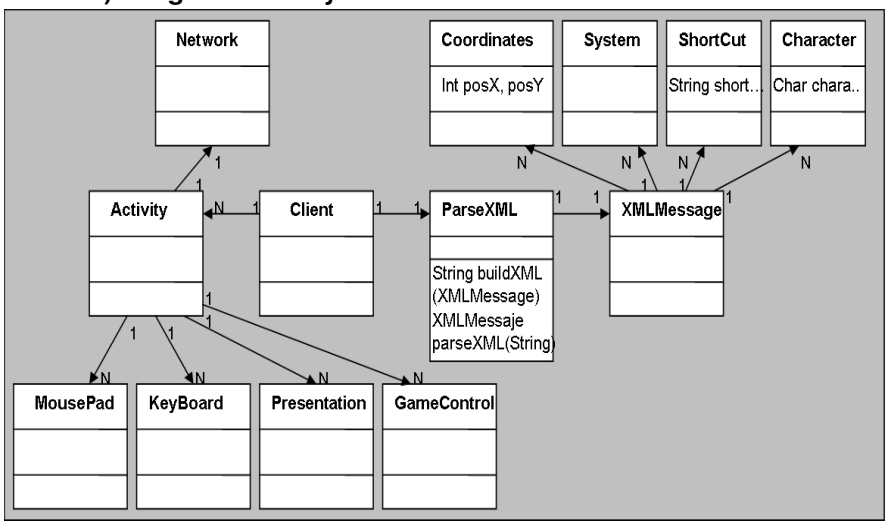
La arquitectura de este sistema operativo está inspirada en componentes de internet, así la interfaz de usuario está diseñada haciendo uso de XML

(Extensible Markup Language), la ventaja de esto es, que sin importar el tamaño de la pantalla la GUI (Graphic User Interfaz) se adaptara a cada uno.

Igual que con Java, Android utiliza una máquina virtual para correr sus aplicaciones, esta es la Dalvik creada por Google, cada aplicación corre en su propio proceso Linux con su propia instancia de la máquina virtual[1].

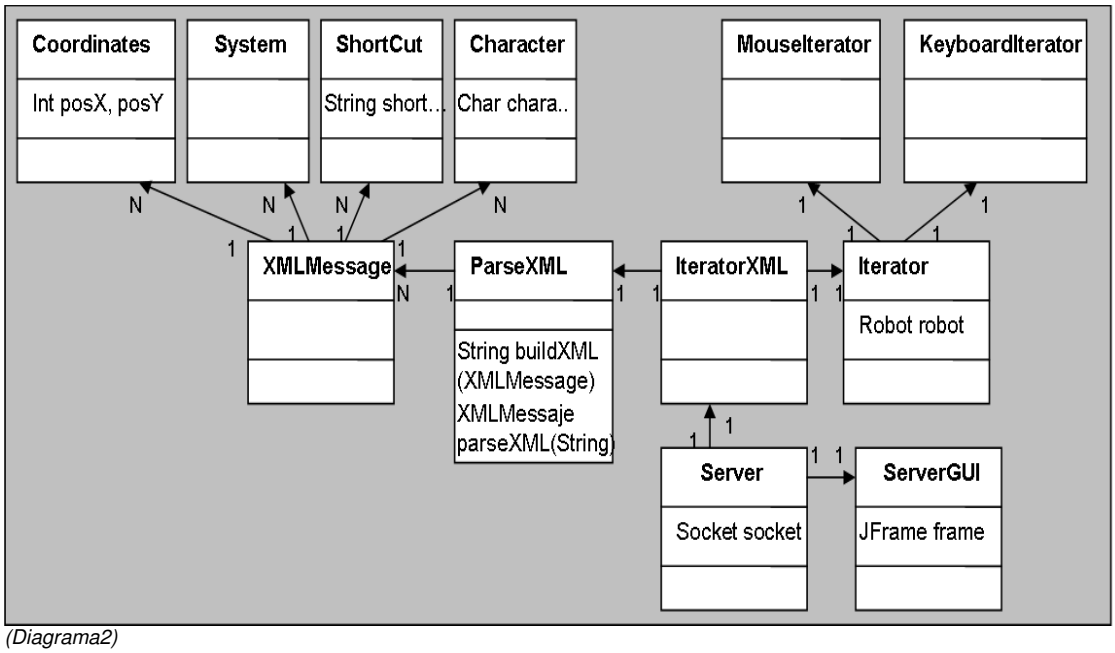
Una vez planteados los objetivos del proyecto se siguieron los siguientes pasos:

A) Diagrama UML y de casos de uso



(Diagrama1)

En este diagrama *(Diagrama1)* se destacan dos de los aspectos más importantes de la aplicación. El protocolo de conexión, la clase Network, y el de comunicación, XMLMessage.
Durante el funcionamiento de la aplicación se recopila información en las distintas actividades. Esta misma es almacenada en las clases que heredan de XMLMessage. Luego, la clase ParseXML convierte estas clases en un String que contiene un XML. Finalmente, el cliente hace uso de la actividad Network para enviar ese String a la aplicación del escritorio.



(Diagrama2)

Como vimos en el *(Diagrama1)* la aplicación móvil enviará un String mediante sockets, conteniendo distintos tipos de información en forma de un XML y será la labor de la clase ParseXML *(Diagrama2)* volver a convertir este XML a una clase equivalente del servidor.

Una vez realizada esta tarea se procede a la clase Iterator, que realiza distintas tareas según la información recibida. Por ejemplo, si recibe un mensaje con coordenadas moverá el puntero del servidor, y si recibe un mensaje con un carácter se presionara dicha tecla en el teclado.

B) Codificación:

Para la codificación del programa de computadora se trabajó sobre Java, JDK (Java Development Kit) 7.[2] Mientras que para la aplicación móvil se hizo uso del android SDK (Software Development Kit), 2.3.3.

Nos decidimos a trabajar con un paradigma orientado a objetos dada nuestra experiencia en este mismo y que es el más difundido entre los dispositivos móviles, así el trabajo en conjunto entre ambos programas se vuelve más simple.

Otros motivos por los que preferimos este paradigma fueron sus facilidades para:

- La reutilización del código, algo de gran valor dadas nuestras aspiraciones de cubrir otro tipo de dispositivos.
- El diseño del programa
- La documentación del código
- Tanto para el mantenimiento como para la expansión del programa, dado que se puede modificar solo el área o la clase que resulte obsoleta, sin que esto resulte en tener que modificar las demás necesariamente.
- Su excelente manejo de datos

Los entornos de desarrollo utilizados en el proyecto fueron el IntelliJIDEA (12.1.2) junto con el Eclipse versión Juno.

C) Control de versión (VCS)

Al estar trabajando en equipo fue preciso llevar acabo un buen seguimiento de los cambios, realizados por cada uno de los programadores.
Decidimos usar GIT[5], este sistema de control de versión de software libre nos permitía, obtener todos los cambios realizados por otro en el proyecto de una manera rápida y sencilla, así como también nos presentaba un informe sobre cambios efectuados en el mismo.

D) Testeo

Para el testeo dispusimos de un grupo de voluntarios, los cuales hicieron uso de la aplicación durante el periodo de un mes, en el cual nos informaron de una serie de errores o "bugs" que no tardamos en resolver, así como también nos proporcionaron nuevas ideas y mejoras para nuestra aplicación.

Dentro de este grupo contamos con usuarios de Linux, Mac y Windows para cerciorarnos del correcto funcionamiento en los sistemas operativos más populares. También contamos con usuarios de distintas versiones de Android, siendo la mayor parte usuarios del 2.3 (Ginger Bread), otros las versiones más recientes como el 4.0 (Ice Cream Sandwich) o 4.2 (Jelly Bean) y en menor medida usuarios de la versión 3.0 (Honeycomb) para tablets[1].

E) Documentación

Tuvimos cuidado de realizar una documentación detallada de cada clase de nuestro proyecto, al trabajar en equipo esto nos ayudó a mejorar la performance, minimizar los errores de implementación y de relación entre nuestras distintas clases.

5. Bases tecnológicas

- La conexión entre ambos programas está realizada mediante Network Sockets. Estos se refieren a un puntero o un puerto de llegada del flujo de datos entre dos dispositivos. En este caso, una computadora y un dispositivo móvil.

  Se lleva a cabo mediante un protocolo de red TCP/IP (Transmission Control Protocol), con el cual identificamos cada uno de los dispositivos. También, un número, que identifica el puerto de salida y entrada de la información remota.

Cuando se trabaja con Network Sockets se implementa una arquitectura cliente-servidor, en la cual el servidor permanece a la espera de que un cliente se conecte. Al realizarse la conexión se procede a la lectura y escritura de mensajes.

- Los mensajes entre ambos dispositivos son enviados haciendo uso de un protocolo XML, facilitado con la librería (XStream 1.4.4)[6].

  Esta elección se debió a que preferimos utilizar un protocolo más genérico, no limitarnos a alguna solución que se redujera solo a Android-Java.

  Este lenguaje fue diseñado por la W3C e igual que HTML consta de tags. Sin embargo, el contenido y formato de estas puede ser diseñado por cada usuario.

  Nuestro programa funciona enviando cierto número de mensajes específicos, identificamos cada uno de estos y los convertimos en convenientes XML, que pueden ser enviados e interpretados por ambos programas.

- El acceso a las opciones de bajo nivel en el servidor, está dado por la clase de Java (AWT.Robot).

- La interfaz de usuario del programa del servidor está diseñada con la librería (Swim).

6. Mejoras y nuevas versiones

Futuras versiones de nuestra aplicación incluirán; El envío de la imagen del escritorio al dispositivo, facilitando así muchas funciones y permitiendo también el acceso remoto. Deberemos desarrollar un buen manejo de la compresión de imágenes para no comprometer la performance. También, la conexión mediante internet y bluetooth, las versiones para otros sistemas móviles y la posibilidad de escoger qué dispositivo será el “Client” y el “Server”.

7. Bibliografía

1. El gran libro de android, Tomás Gironés Jesús, 3 edicion, MARCOMBO, S.A.

2. S. Microsystems, «JDK 7 Project,» [En línea]. Available: http://jdk7.java.net/.

3. The Eclipse Foundation, «Eclipse Juno»[En linea]. available: http://www.eclipse.org/downloads/.

4. « IntelliJIDEA 10.5,» [En línea]. Available: http://www.jetbrains.com/idea/download/.

5. GIT [En linea]. Available: http://git-scm.com/.

6. XStream [En linea] Available: http://xstream.codehaus.org/.

7. TeamViewer [En linea] Available: http://www.teamviewer.com/es/index.aspx.

8. Remote Mouse [En linea] Available: https://play.google.com/store/apps/details?id=com.hungrybolo.remotemouseandroid