Evaluación de JavaME

Gabriel Centurion
gccomputos@gmail.com

Maximiliano Felix
maxi.felix@gmail.com

RESUMEN

El objetivo del siguiente trabajo es investigar, utilizar y evaluar el estado del arte de JavaME (Java Micro Edition) [1], como plataforma para desarrollo de aplicaciones para celulares.

En la actualidad los dispositivos móviles (celulares) forman parte de nuestra vida cotidiana. No solo como teléfonos móviles sino además como cámaras de fotos, filmadoras, calculadoras, despertadores, relojes, GPS entre otros.

Si a lo anterior se le agregan las posibilidades que brinda el acceso a Internet desde el propio celular las posibilidades se multiplican.

Dentro de este marco se inició un proyecto que permitiera investigar la mayor cantidad de tecnologías aplicables en un dispositivo móvil. Para ello se decidió desarrollar un juego de cartas llamado "Truco" (en modalidad uruguaya) el cual brinda soporte multijugador tanto por bluetooth como por web services, utilizando JavaME como plataforma de desarrollo.

Como principal conclusión se destaca que JavaME es una plataforma aceptable para el desarrollo de aplicaciones móviles y en conjunto con sus paquetes opcionales se convierte en un poderoso framework, que tiene entre sus principales ventajas la de que permite un alto grado de abstracción del dispositivo evitando de esta manera gran parte de ajustes de software según el fabricante.

Palabras Clave

JavaME, Web Services, Bluetooth, Multijugador, Truco, Juego de Cartas, Celular.

1. INTRODUCCIÓN

JavaME es un framewok de Java para el desarrollo de software para dispositivos de recursos limitados. Al igual que JavaSE ejecuta sobre una maquina virtual (JVM) que en el caso de los dispositivos en cuestión se conoce como KVM (Kilo Virtual Machine).

Lo interesante de la tecnología en cuestión es que cada fabricante tiene su propia KVM y mediante estándares que deben de cumplir todos los fabricantes es que se logra que las aplicaciones realizadas en Java ejecuten en diferentes dispositivos de diferentes compañías con diferentes sistemas operativos que provean soporte Java.

Por otra parte hay proveedores de librerías para solucionar distintos temas. A modo de ejemplo para interpretar XML se tuvo la posibilidad de utilizar no menos de cinco librerías pero al momento de prototipar recién la tercera funciono como prometía. Esto hace más interesante y a la vez más compleja la plataforma. Puesto que no solo hay que encontrar los paquetes que se ajusten a nuestras necesidades sino que los mismos sean compatibles con el resto de los paquetes, que su peso dentro del instalador amerite el incluirlo y que su funcionamiento sea lo suficientemente aceptable.

Además de probar las distintas tecnologías a aplicar, se planteó desarrollar el caso de estudio en varias capas abstrayendo incluso la lógica del juego de la misma capa lógica, para de esta manera verificar si es posible desarrollar en JavaME software reutilizable, en contraposición a la perdida de performance por la comunicación entre capas. Se notó la posibilidad de ello como también que se puede trabajar como si se estuviese desarrollando en JavaSE salvo el Api reducido, cuidando el mantener la performance a la hora de hacer algoritmos complejos y evitando cálculos innecesarios. Un ejemplo de evitar algoritmos complejos se planteó al momento de decidir si utilizar una imagen para representar las cartas o dibujarla utilizando Java 2D , se consideró que es más performante utilizar imágenes sobre todo mediante Sprites que hacer que el procesador dibuje cada vez cada carta.

Al tratarse de un juego, tomando en cuenta que las pantallas de los móviles cada vez son más dispares y con distintas resoluciones, se evitó el uso de canvas. Se adoptó el framework LWUIT el cual sería para JavaME lo que es Swing para JavaSE. Dicho framework permite el uso de layouts (entre otros) los cuales se adaptan a cada pantalla manteniendo los componentes gráficos ubicados en la posición deseada, independientemente de la resolución (dentro de lo posible).

Para abarcar cierta gama de celulares y que las tecnologías no fuesen una limitante se decidió la utilización en CLDC 1.1 que es la última versión libre hasta el momento (también existe una versión mejorada llamada hotspot 1.1.3) MIDP 2.0 que es la penúltima versión y LWUIT 1.4 que es la última versión.

Este trabajo se plantea como tarea de la edición 2010 de la Materia taller de Sistemas de Información 4.

El resto del documento se organiza de la siguiente manera. La sección 2 presenta el Marco Conceptual en donde se explican los principales conceptos tratados. En la sección 3 se describe el caso de estudio utilizado para la investigación tecnológica. En la sección 4 se detalla la solución planteada para el caso de estudio indicado en la sección 3. En la sección 5 se describe la arquitectura del sistema y como se recorren las distintas capas en un evento tipo. En la sección 6 se mencionan los productos y herramientas utilizados, dando una breve evaluación de los mismos. Además se presentan los problemas encontrados que más se destacan. En la sección 7 se evalúa la solución alcanzada. En la sección 8 se describe las distintas etapas del desarrollo del proyecto. Por último, en la sección 9 se presentan las conclusiones y el trabajo a futuro.

2. MARCO CONCEPTUAL

En esta sección se describen varios conceptos que son necesarios para la comprensión del artículo.

2.1 JavaME

Java Micro Edition es una plataforma robusta y flexible para el desarrollo de aplicaciones para dispositivos móviles. En si, una familia de especificaciones que definen varias versiones minimizadas de la plataforma Java.[1]

2.2 Java2D

API para dibujar gráficos en dos dimensiones usando el lenguaje de programación Java. El API 2D de Java permite dibujar líneas de cualquier ancho. Permite además rellenar formas con gradientes y texturas. Es posible mover, rotar, escalar y recortar texto y gráficos. Brinda soporte para componer texto y gráficos solapados. A las imágenes se les puede realizar filtros, como blur o recortado utilizando dicha API. [3]

2.3 JSR

Java Specification Request

Son el resultado de la Java Community Process. La cual es un proceso formalizado que permite que las partes interesadas puedan participar en la definición de nuevas características de la plataforma Java. Como resultado de ese proceso se genera una serie de documentos formales que indican el estándar a adoptar por los involucrados. Cada JSR (una vez aprobado o final) brinda una implementación de referencia libre de la tecnología en código fuente además de los documentos formales.[4]

2.4 BlueTooth

Tecnología que permite mediante ondas de radio la comunicación entre dispositivos mediante un enlace por radiofrecuencia. [5].

2.5 WS

Web Services son un conjunto de protocolos y estándares que sirven para intercambiar datos entre aplicaciones. [6]

2.6 XML

eXtensible Markup Language es un metalenguaje extensible de etiquetas. [7]

2.7 CLDC

Connected Limited Device Configuration

Esta configuración está diseñada para dispositivos con conexiones de red intermitentes, procesadores lentos y memoria limitada como son teléfonos móviles, asistentes personales (PDAs), etc. [8]

Está orientado a dispositivos que tengan procesadores de 16 bit/16 MHz o más. Memoria de 160-512 KB disponible para la plataforma Java. Alimentación limitada, a generalmente basada en batería. Conectividad a algún tipo de red limitado en su ancho de banda. En particular CLDC 1.1 (JSR 139) es una revisión de la especificación CLDC 1.0 que incluye nuevas características como son punto flotante o soporte a referencias débil, junto con otras mejoras. Es compatible con versiones anteriores y sigue soportando dispositivos pequeños o con recursos limitados.

2.8 MIDP

Mobile Information Device Profile. Proporciona un perfil que se apoya en el CLDC y brinda las clases y paquetes necesarios para el desarrollo de nuestras aplicaciones en celulares.[9]

En particular El API de MIDP 2.0, se compone de:

- Paquete de ciclo de vida de las aplicaciones llamado javax.microedition.midlet. El cual permite a las aplicaciones MIDP (midlets) interactuar con el entorno sobre el cual la aplicación se está ejecutando.
- El paquete de Interfaz de usuario llamado javax.microedition.lcdui. El cual consta de una interfaz de usuario basica.
- Paquete de Juegos llamado javax.microedition.lcdui.game. El cual proporciona una serie de clases que permiten construir juegos como ser el manejo de Sprites.
- Paquete de Red llamado javax.microedition.io´, que brinda soporte de red basándose en CLDC.
- Paquete de Clave Pública llamado javax.microedition.pki se utiliza para el manejo de certificados que autentiquen información proveniente de conexiones seguras.
- Paquete de Media javax.microedition.media es compatible con la especificación Mobile Media API (MMAPI). MMAPI extiende la funcionalidad de JavaME proporcionando audio, video y otras características multimedia. Es un paquete opcional, simple y ligero, que también permite acceder a los servicios multimedia nativos. Complementando el anterior javax.microedition.media.control define los tipos de control específicos que pueden ser usados en el reproductor de la API Media.
- Paquete de Persistencia llamado javax.microedition.rms el cual se explica posteriormente en el punto RMS
- Y los paquetes java.lang y java.util reducidos.

2.9 MIDLET

Se denomina así a las aplicaciones Java realizadas usando MIDP como referencia. Un MIDlet es una aplicación que va a ejecutarse en un dispositivo móvil. La estructura de un MIDlet consta de tres métodos obligatorios startApp(), pauseApp() y destroyApp(boolean unconditional) que son llamados en iniciar pausar y terminar una aplicación respectivamente. En dichos métodos se puede guardar un contexto inicializar la aplicación o cambiar de estado la misma si es necesario. [10]

2.10 RMS

Tecnología para manejo de base de datos (o mejor dicho simulación de base de datos) MIDP proporciona un mecanismo para almacenar datos de forma persistente y poder recuperarlos posteriormente.

El mecanismo se llama Sistema de almacenamiento de registros (Record Management System, RMS), y se basa en el modelo simple de base de datos orientada a registros. [11]

2.11 Layout

Diagramación o maquetación. Es la descripción que indica donde y como van ubicados los componentes en la interfaz de usuario. [12]

2.12 LWUIT

Lightweight User Interface Toolkit es una librería de componentes gráficas para JavaME. Aporta componentes gráficos más ricos que las librerías estándares para el desarrollo de aplicaciones móviles además de efectos gráficos, transiciones, efectos 3D, manejo de resources empaquetados entre otros. [13]

2.13 Sprites

Conjunto de imágenes dentro de una misma imagen que mejoran sustancialmente la performance en el manejo de imágenes. Utilizados en juegos desde la época de las “maquinitas” y han ido evolucionando en la historia de los videojuegos. [14]

2.14 OTA

El proceso por el cual un usuario instala una aplicación en un dispositivo móvil se llama en inglés "provisioning", Over the Air provisioning (OTA) refiere al proceso por el cual el usuario descarga en forma inalámbrica la aplicación en el dispositivo y la instala. Los emuladores tienen una simulación de esto, tomando como entrada los archivos JAD y JAR generados para la aplicación. [15]

2.15 Push Registry

Mecanismo introducido por MIDP 2.0, permite ejecutar MIDlets de forma automática, sin la iniciación del usuario. Para utilizarlo simplemente se declara en el JAD, el MIDlet que se va a iniciar, junto con una URL, y la clase Java del MIDlet. Cuando llegue una conexión a la URL declarada, el MIDlet será instanciado automáticamente por el dispositivo. [16]

2.16 JAD

Java Application Descriptor es un archivo en el cual se registran los midlets ejecutables que tiene un jar con su mismo nombre. También es donde se declaran los permisos que la aplicación requiere para utilizar algunos recursos del dispositivo, entre otras cosas. [17]

2.17 JAR

Java Archive, es un archivo java que tiene empaquetadas una serie de clases, además permite ejecutar aplicaciones escritas en lenguaje Java. [18]

2.18 Marge

Librería para el manejo de Bluetooth que facilita el uso de JSR82. [19]

2.19 Truco

Juego de cartas con baraja española de origen árabe.

En su variante uruguaya se juega con "muestra", que es una carta que se coloca a la vista bajo el mazo, luego de repartir las cartas e incorpora al juego cinco cartas del mismo palo de dicha muestra que se denominan "piezas" y que son de mayor a menor: dos, cuatro, cinco, caballo y sota. Estas piezas son las de mayor valor, por encima del as de espadas. Al caballo y sota de la muestra se los llama "perico" y "perica" respectivamente. En el caso de quedar de muestra alguna de esas "piezas", su lugar en el juego lo asume el rey del mismo palo ("alcahuete"), de tal forma que las cinco piezas siempre estén en juego. [2]

3. DESCRIPCIÓN DEL PROBLEMA

El problema planteado consiste en el diseño y construcción de un juego de cartas "Truco", el cual permitirá testear el estado del arte de diferentes tecnologías involucradas en la plataforma JavaME. El mismo permitirá que dos o más usuarios jueguen entre sí mediante sus teléfonos móviles ya sea conectados a través de bluetooth o de internet (mediante WS).

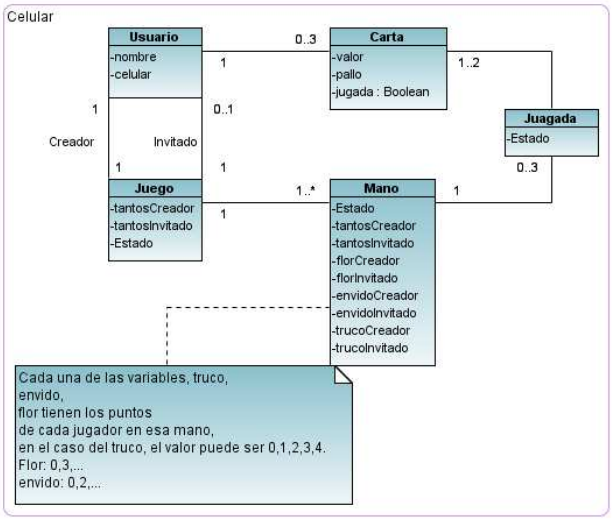
Las funcionalidades que debe proveer el sistema son:

- Crear un nuevo juego, el cual tomará como datos el nombre del usuario que se quiere invitar, y permitirá enviar un SMS para avisarle de la invitación.
- Soporte para editar y guardar la configuración de la aplicación entre la cual se encuentra: elegir la forma de comunicación, en caso de ser Internet se permitirá al usuario cambiar la URL del servidor. Seleccionar el nombre de usuario a utilizar en los juegos.
- Envío de SMS al usuario invitado desde la aplicación.
- Ejecución automática la aplicación en el celular invitado al llegar un SMS de invitación.
- En lo concerniente al juego propiamente dicho debe proveer reparto de cartas, jugar una carta, llevar tanteador, “gritar” truco, retruco, vale cuatro, quiero, no quiero, envido, flor y todas sus variantes. Irse al mazo, etc.
- Soporte para jugar directamente entre dispositivos, usando la tecnología de comunicación bluetooth.
- Soporte para jugar a través de un servidor web, que se encontrará en Internet, el cual provee ciertos WS para envío y recepción de mensajes, entre los usuarios de un juego.
- Soporte multi-jugador, de dos o cuatro usuarios, jugando en equipos de uno o dos personas respectivamente.
- Reproducción de sonidos de acuerdo a los diferentes eventos del sistema.

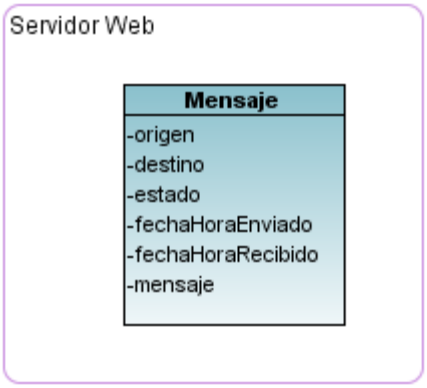
4. SOLUCIÓN PLANTEADA

La solución está compuesta de dos aplicaciones, una de las cuales correrá en los dispositivos móviles, y otra será un servidor web que proveerá los servicios a ser consumidos por la aplicación móvil si esta lo requiere.

La aplicación móvil tendrá el siguiente modelo de dominio, que comprende lo necesario para modelar un juego de truco.



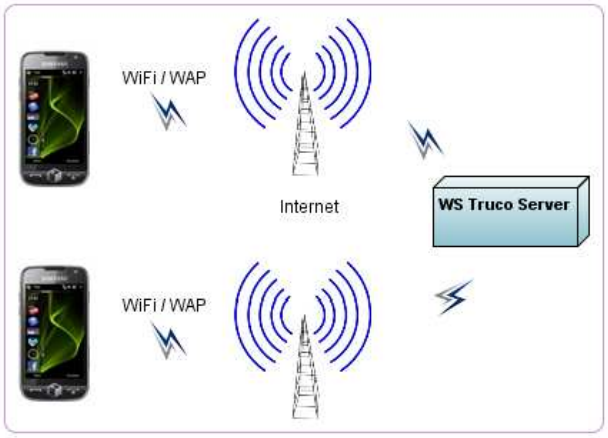
El servidor web simplemente se limitará a recibir y enviar mensajes cuando sea consultado por mensajes sin leer. Para el servidor estos mensajes son transparentes, no guarda ningún tipo de información respecto del juego entre dos usuarios. Por tanto el modelo de dominio del servidor es muy acotado, solamente los mensajes a ser procesados.



El problema plantea al menos dos tipos de comunicación entre los dispositivos que correrán la aplicación. Esta se expresa en las siguientes imágenes, que ilustran como será la comunicación durante el juego.

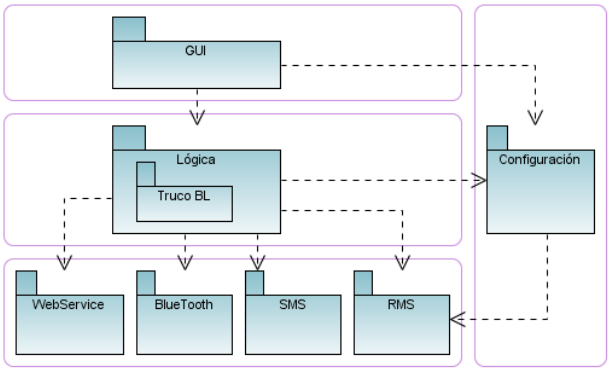
Juego a través de BlueTooth.

Juego a través de Internet.

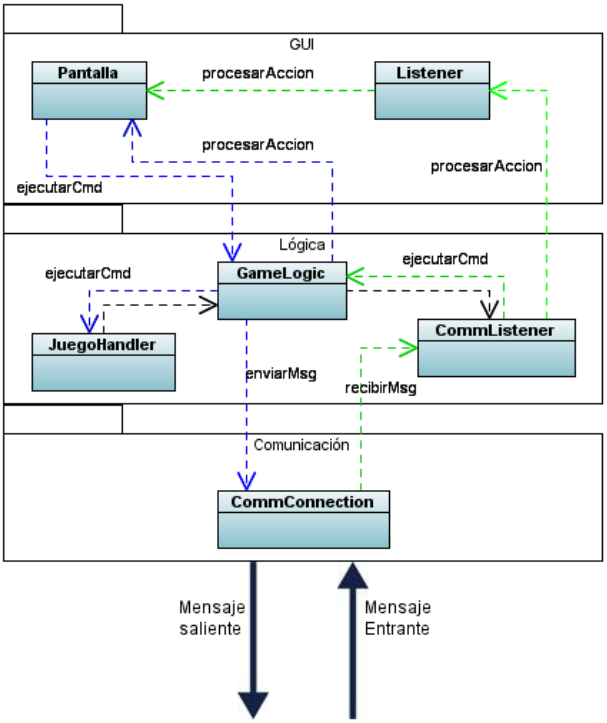


5. ARQUITECTURA DEL SISTEMA

La arquitectura propuesta es básicamente de 3 capas, presentación, lógica y acceso a recursos como ser comunicación y persistencia. Además existe una cuarta capa auxiliar que es transversal a todas las capas y se le denomina de configuración.
La capa superior corresponde a la presentación donde se alojará tolo lo relativo a componentes gráficos. La capa de presentación tiene acceso a la capa lógica y a la capa de configuración. En la segunda capa se encuentra la lógica de negocio, dentro de la misma estará la lógica del juego.
De esta manera se pretende aislar la lógica del juego, para que cambiando la misma y la capa de presentación se pueda reutilizar el resto para desarrollar un juego distinto.
La capa lógica tiene acceso a la capa de recursos la cual provee de persistencia y comunicación. La capa de recursos se divide en dos. El acceso a datos mediante la comunicación entre dispositivos la cual puede ser a través de bluetooth, web services o SMS, y la segunda parte es la persistencia de datos en el dispositivo utilizando la librería de RMS.



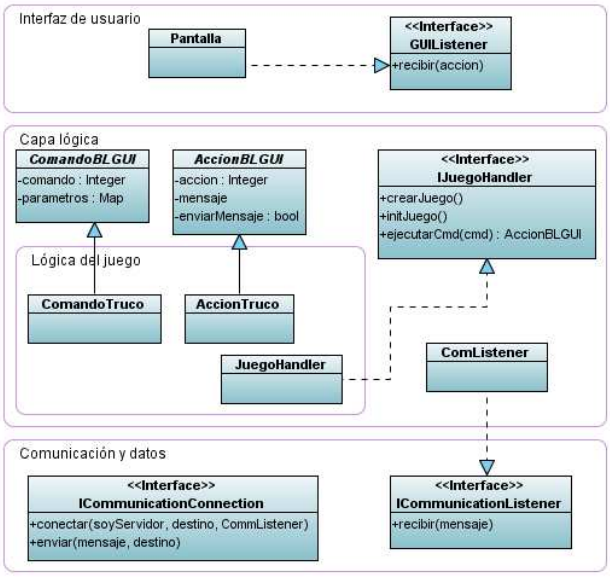
En las siguientes líneas se explica funciona el sistema de mensajes entre celulares a través de las diferentes capas de la aplicación.



En el diagrama anterior se puede identificar dos flujos de datos, el que está en azul (gris oscuro en mono tono) corresponde a las acciones realizadas por el usuario en la Gui, tiene como punto inicial o de partida del flujo la pantalla, apretar un botón u otra acción del usuario provocará que esta, ejecute un comando en la lógica del juego, que procesará el comando enviado por el usuario y retornará una acción a procesar por la Gui, esta acción podría eventualmente provocar que se envíe un mensaje al otro dispositivo. Por lo tanto el flujo azul (gris oscuro en mono tono) puede tener dos puntos de fin, un cambio en el estado de la pantalla mostrada al usuario y un mensaje saliente hacia el otro dispositivo.

Por otro lado se encuentra el flujo marcado en verde (gris claro en mono tono) el cual ocurre cuando llega un mensaje del otro dispositivo, este será el punto inicial. El mensaje es recibido por la capa de comunicación y enviado al Listener una capa más arriba. El mismo es transformado en un comando y es pasado a la lógica del juego para que lo procese y retorne una acción que será procesada luego por la Gui. Dicha acción será procesada a través de su GuiListener para indicarle al usuario que ha ocurrido un evento. Este flujo en algunos casos podría también enviar un mensaje al otro dispositivo en respuesta al mensaje recibido.

Profundizando en cómo se componen las interfaces que brindan la abstracción entre las diferentes capas. El siguiente diagrama muestra las interfaces ubicadas en cada capa y sus principales operaciones así como donde se encuentran quienes las implementan.



Tal vez lo más interesante a resaltar del diagrama anterior, es como se aísla la lógica del juego del resto de la capa lógica. Esto se logra mediante la implementación de la interfaz IJuegoHandler, y la extensión de las clases abstractas AccionBLGUI y ComandoBLGUI. Las mismas son las responsables de contener la información que viaja entre la capa lógica y la GUI. En el caso de estas dos últimas clases, tanto el *comando* como la *acción* son enteros, cuyo valor y significado es dado por las clases hijas, *enviarMensaje* es utilizado por JuegoHandler para indicarle al GameLogic que el resultado de ejecutar un comando, debe enviar un mensaje al otro dispositivo. JuegoHandler no solamente implementa la interfaz, sino que también es el encargado de mantener el modelo de datos durante el desarrollo del juego, logrando así una abstracción total de la lógica del juego, de la capa lógica de la aplicación.

Lo explicado en los párrafos anteriores se refiere solamente a la aplicación a correr en los dispositivos móviles, pero también se encuentra la arquitectura del servidor web que provee los web services consumidos por la aplicación móvil. Este será implementada usando Grails, un Framework de desarrollo web Dicho servidor será muy simple tal cual plantea el modelo de datos, limitándose a almacenar los mensaje recibidos sin realizar trabajo alguno sobre ellos.

6. IMPLEMENTACIÓN

En esta sección se brindan más detalles sobre la implementación del sistema, centrándose en la evaluación de las tecnologías involucradas, así como los problemas encontrados y su forma de resolverlos.

6.1 Productos y Herramientas

En la siguiente tabla se muestran las diferentes tecnologías utilizadas y una breve evaluación de las mismas.

Tabla 1. Evaluación de Productos y Herramientas

Producto	Puntos Fuertes	Puntos Débiles	Evaluación General
JavaME	Curva plana de aprendizaje	Muy limitado en cuanto al soporte de	En general es una plataforma

	al ser basada en Java, compatibilidad con muchas librerías Java, bondades Cross-Platform de Java. Gran variedad de tecnologías opcionales que la hacen muy completa.	librerías estándar de Java. Gran cantidad de los paquetes de interacción son opcionales, lo que limita la cantidad y variedad de dispositivos en los que puede correr una aplicación.	muy completa para desarrollo de aplicaciones móviles, pero hay que tener cuidado, porque el concepto "Write once, run everywhere", no es cien por ciento aplicable a esta tecnología.
Wireless Toolkit 2.5.2	Muy completo, estable y útil. Se integra con Eclipse.	No se encontraron.	Conjunto de herramientas fundamental para el desarrollo, pues trae los emuladores, que son muy eficientes para el testeo de las aplicaciones.
Eclipse	Velocidad, fiabilidad, adaptabilidad	No se encontraron.	Excelente IDE para desarrollo de aplicaciones Java en general, y muy bueno para desarrollo de aplicaciones JavaME.
Plugin EclipseME	Brinda lo mínimo necesario para el desarrollo y más.	Le falta un editor gráfico de pantallas. Discontinuado. O al menos no soporta las últimas versiones de Eclipse con su versión release. La última noticia al respecto e s de Junio del 2009.Se instala aunque no sea compatible	Indispensable para el desarrollo, muy bueno en el estado que está pero le falta un editor, que acelere el proceso de desarrollo de la GUI. No fue la mejor elección este plugin esta discontinuado y se siguió con otro que es parte de la distribución

		con la versión de Eclipse se utiliza.	de Eclipse llamado Pulsar.
Marge[19]	Facilita el uso de JSR82. Estable y liviano.	No se encontraron.	Revisando un par de ejemplos y teniendo claro el manejo de listeners se puede realizar un prototipo en pocos minutos.
LWUIT	Se puede implementar con conocimientos de Swing rápidamente. Provee una serie de transiciones, efectos y manejo de recursos muy interesante.	En algunos casos no refresca directamente los componentes.	El framework se ve muy maduro, salvo algún work-arround no presentó mayores problemas. Da la impresión que para sacarle el mejor provecho hay que dedicarle más investigación a ciertas operaciones que manejan ciertos estados del mismo Framework, que se utilizaron sin mucha profundidad.
Sprites	En una imagen se tienen todas las necesarias. Se accede menos al storage para levantar imágenes por lo que mejora la performance.	Hay que dedicarle un poco de tiempo a ubicar correctamente la imagen deseada.	El manejo provee java sobre sprites es muy aceptable.
Mobile Media API	Abstrae la utilización de la Media del celular	En los emuladores en algunos casos sonó	Se utilizó para el sonido y resulto fácil

	como ser audio, video, cámara.	con ruidos pero en los teléfonos se comporto correctamente	su uso.
XParse	Liviana estable y sencilla	No se encontraron.	La primera de tres librerías que cumplió lo que prometió, liviana sencilla se adapto bien a las necesidades de la aplicación.
RMS	Fácil implementación.	Muy limitado, solo guarda registros de bytes consecutivos.	Es interesante y muy necesario tener cierto nivel de respaldo de datos, pero con esto solamente, es impensable construir una aplicación compleja de manejo de datos
SMS API	Mucha documentación, fácil de utilizar y de entender.	No se encontraron.	Al igual que otras formas de conectividad, es muy fácil de utilizar, lo cual es algo muy bueno.
WS API	Muy fiable, combinado con WTK agrega simplicidad.	Código generado es poco entendible.	Muy buena herramienta para el consumo de WS, no lo es así para la exposición de servicios
Grails-Xfire	Simple, fácil de utilizar, rápido desarrollo.	No se encontraron.	Máxima productividad, facilidad de uso y compatibilidad con JavaME.
Push Registry	Muy simple de utilizar y de entender, solo una declaración en el JAD	No es fácil de testear en el emulador, requiere levantar la aplicación en forma especial.	Excelente para interactuar con el dispositivo y brindar mayor usabilidad al usuario.

6.2 Problemas Encontrados

En cuanto a los problemas dentro de la plataforma se destaca soporte limitado de Java SE. Esto refiere a que no todas las librerías y tipos de Java se encuentran soportados en JavaME. En algunos casos como por ejemplo BigDecimal, no se encuentra incluido en la plataforma. De las colecciones del paquete java.util, solo se encuentra HashTable. El soporte de java.lang.Math, es acotado a un subconjunto de métodos. Por ejemplo el método random, no lo soporta. Este método resulto necesario para repartir las cartas y se logró sustituir por otra librería que proveía para el caso la clase Random y el método nextInt. Sobre los métodos estáticos de los tipos Boolean, Long se puede mencionar que algunos de estos métodos como por ejemplo Boolean.valueOf, no está soportado. Para estos casos se tuvo que utilizar la comparación con el string “true”.

Sobre los casos antes comentados se destaca que los errores se dan en tiempo de ejecución, siendo más costosa en tiempo su resolución.

Sucedieron problemas con el emulador. Para ejecutar la aplicación en el dispositivo móvil a través de SMS, hay que utilizar Push Registry. Este sólo se activa en el emulador cuando se ejecuta vía OTA (over the air), simulando su instalación en el dispositivo. Para que la aplicación no se ejecute automáticamente hubo que agregar un MIDLET DUMMY, el cual provoca que antes de levantar la aplicación, se muestre una lista con los MIDLETS, de esta forma se pudo testear el levantar la aplicación a través de SMS.

Sobre inconvenientes con EclipseME se puede mencionar que fue necesario instalar la versión 1.8.0 la cual es de test. Esto se realizó por compatibilidad con la versión de Eclipse que se utilizó. La última versión release la cual es la 1.7.9 no funciona con la versión que se utilizo. La misma hace que no ejecute el programa en el emulador. Fue muy costoso en tiempo conseguir la solución puesto al respecto, puesto que el plugin se instala correctamente y no informa el estado de la compatibilidad.

Se suscitaron inconvenientes con los permisos de usuario. Para utilizar recursos de red, bluetooth, SMS se deben declarar permisos especiales en el archivo JAD de la aplicación. En caso de no indicar dichos permisos dará un error por la falta de los mismos en tiempo de ejecución.
En especial al utilizar recursos de conectividad, se debe instanciar las nuevas conexiones en hilos diferentes. Pues el emulador y casi todos los dispositivos reales, lanzan una advertencia al usuario, el cual debe presionar un botón para continuar. Si todo ocurre en el mismo hilo, la aplicación quedará bloqueada.

Para utilizar el generador de archivos resources que provee LWUIT se requiere modificar manualmente, ciertos archivos ANT. Para de esta manera poder ingresar, quitar o modificar recursos y tome en cuenta los cambios requeridos. En los ejemplos esos archivos están basados en Netbeans y es bastante costoso adaptarlos aunque es posible. Lo que sucede es que LWUIT trae su propio editor de resources pero no es nada

práctico modificar desde el editor. Por lo que al tener estos archivos ANT llamando a una clase desde los mismos se logró generar los resources necesarios.

Continuando con LWUIT hubo además ciertos problemas al querer refrescar la pantalla con nueva información. Pues LWUIT tiene su propio manejo de hilos, el cual no es muy claro en su funcionamiento. Obligando por ejemplo a refrescar todo un contenedor para poder visualizar el nuevo contenido.

Al instalar la aplicación en dispositivos reales se noto una serie de diferencias. Por ejemplo existe una demora de hasta 15 segundos en establecer la conexión por bluetooth. Lo que no se pudo determinar por falta de debugger contra la KVM del mismo dispositivo es si en realidad es un problema del framework MARGE de los dispositivos en si o el propio soporte de bluetooth el que exista dicha demora. Continuado las pruebas en dispositivos reales, se encontraron los siguientes inconvenientes. En un celular Nokia 5200, si bien según la especificación del fabricante debería funcionar, cuando se colocaron los archivos en el mismo, no dice "Aplicación Inválida" y no deja instalarla de ninguna manera. Por lo tanto no se pudo probar en este dispositivo. Habiendo instalado la aplicación en un Samsung Omnia 2 y en un Sony Ericsson Vivaz sin dificultades, se encontró que la resolución de la pantalla de dichos móviles es mucho mayor a la de los emuladores, quedando las cartas muy pequeñas para estos dispositivos. Se podría solucionar teniendo más de un juego de cartas con distintos tamaños dependiendo de la resolución del móvil. Dicha resolución se conoce a través de LWUIT.

Continuando con el tema del debugger cabe mencionar que el mismo no funcionó. Se supone que se debe a la versión Eclipse y del plugin que se utilizó. Debido a ello se implemento un sistema de logueo similar a Log4J a grandes rasgos. El sistema de logueo permitió continuar el hilo de ejecución y conocer donde se existían errores.

Si bien no se tuvo problemas con RMS, si se tuvo con la combinación de este con el emulador, pues el emulador utiliza como base una carpeta que genera en cada ejecución, lo que provoca que lo guardado con RMS, a la siguiente ejecución no se encuentre. Para resolverlo se agregó una inicialización que si no se encuentran presentes los datos por defecto, se guardan.

7. EVALUACIÓN DE LA SOLUCIÓN

Se considera que la aplicación no está completa, por lo tanto no deja de ser un prototipo. Se dice que no está completa porque falta implementar casos de uso importantísimos para el desarrollo del juego Truco. Los mismos por temas de tiempo fueron dejados de lado. Además de ser funcionalidades que no aportan al objetivo de la materia que es la investigación tecnológica.

Enumerando los casos de uso dejados de lado, se encuentran:

- Soporte para 4 jugadores.
- El envido, real envido, falta envido.
- La flor, contra flor, etc.
- "A ley de juego todo dicho."

Por otro lado se pueden indicar las bondades de la aplicación desarrollada, como ser las que brinda la arquitectura usada. Por ejemplo, aislar todo lo relacionado a recursos de conectividad en una capa y la elección de estos, de la capa lógica, permitió desarrollar gran parte de la aplicación utilizando conectividad bluetooth. Posteriormente se paso a utilizar WS sin necesidad de retocar ni una línea de código. También se debe mencionar que se logró uno de los objetivos iniciales, que era aislar por completo la lógica del juego propiamente dicha, de las capas inferiores. Lo anterior permite una total reutilización de la capa de acceso a datos y la capa lógica. Así como debe re implementarse la GUI y la lógica del juego, para tener un nuevo juego funcionando.

Pero la arquitectura también tiene sus puntos débiles, la excesiva abstracción en los niveles inferiores complica la lógica del juego, que debe resolver todo. Por otro lado el tener dos puntos de entrada diferentes también hace la implementación muy meticulosa para no cometer errores por no considerar casos particulares. Por otro lado se tienen algunas decisiones de implementación que obligaron a poner lógica especial en la GUI. Las cuales no permitieron la reutilización de código para atender las mismas acciones. Esta decisión fue que en el caso del creador del juego las respuestas a un comando se retornan en la misma llamada a la lógica. Otra decisión de diseño que puede discutirse sus ventajas y desventajas, es que solo el creador del juego tiene los datos del juego, el invitado se limita a pedir los datos que necesita, esto pareció los más acertado al principio porque evita los problemas de tener sincronizados los datos en ambos dispositivos. Pero al pasar el tiempo se noto que obliga a que cada acción en el invitado termine con un mensaje al creador, y por tanto agrega demoras en la actualización de los datos mostrados al usuario. Así como también muchos mensajes con información vital yendo y viniendo cuando podrían ser simples mensajes de sincronización. A esto último debe agregarse que la aplicación no tiene tolerancia a fallas desde el punto de vista de pérdida de mensajes en la comunicación.

Uno de los objetivos iniciales era cubrir el caso de juego entre 4 personas, en equipos de 2 contra 2. Esto fue descartado por razones de complejidad, respecto a cómo cambian las reglas del juego entre 2 personas y 4 personas, aún así evaluando la solución alcanzada se puede decir que todavía es complejo alcanzar el objetivo, para lograrlo se deben considerar serios cambios a nivel de la lógica del juego y la GUI. Aunque no se puede descartar cambios más profundos en la arquitectura.

En general se considera que la aplicación alcanzada es muy buena en cuanto que cumple perfectamente con los requisitos para probar a fondo las tecnologías involucradas. Como punta pie inicial para completar al menos el juego de a dos personas es excelente. Solo basta agregar los casos de uso dejados afuera, que no son muy diferentes al caso de uso truco ya implementado.

8. DESARROLLO DEL PROYECTO

Después de una primera investigación preliminar de la plataforma en general, se paso a un estudio meticuloso de las tecnologías a utilizar para implementar la aplicación.

Se procedió con la investigación de las tecnologías y se implementaron prototipos aislados para cada una de ellas. Una vez entendidas se paso a la implementación del esqueleto del sistema.

En los primeros pasos de la implementación, se utilizó NetBeans como IDE de desarrollo pues el editor gráfico de pantallas y el flujo entre estas, es muy bueno. Pero NetBeans tuvo que ser descartado pues su gran ventaja frente a Eclipse, el editor gráfico no es aplicable a LWUIT, que es la librería grafica que se decidió utilizar. Sin editor gráfico en ninguno de los dos IDEs, se decidió por Eclipse pues se consideró más ágil y estable para trabajar, además de transparente en su manejo de los distintos aspectos que hacen al desarrollo y la codificación. También se debió descartar casi todo el código generado por NetBeans porque el mismo utilizaba código propio de NetBeans, resultando muy costoso adaptarlo a Eclipse. De nuevo en cero se procedió a la implementación del esqueleto a mano, tomando como ejemplo el prototipo de LWUIT implementado en la etapa previa.

Con el esqueleto constituido se implementó la capa de comunicación y datos, por separado se hizo bluetooth y WS, con la base construida se pasó a implementar la GUI y la lógica. En esta parte se hicieron notorios errores de diseño en la capa de comunicación que hacían perder abstracción de la capa con respecto a las capas superiores. Se tuvo que re-diseñar e implementar los cambios, ahora si por buen camino se siguió con las capas superiores.

La lógica del juego y la GUI fueron implementadas en paralelo de forma coordinada. Implementando el soporte para cierto caso de uso en la lógica y en la GUI al mismo tiempo. Esta implementación coordinada dejó además los casos de test listos para ir desarrollando y testeando paso a paso, sin tener que invertir más tiempo en implementar tests separados. Como Mocks y Drivers.

Primero se implementó crear un juego, unirse a un juego y repartir las cartas, posteriormente jugar una carta, y ver quién gana esa jugada, esto fue complejo, pues a la complejidad del cálculo de los valores de las cartas, según la muestra, también se agregó que el turno, o sea a quien le toca jugar, varía según si el mano gana la jugada o no. Una vez resuelto el juego de las cartas y asignar los puntos cuando se termina la mano, se pasó a implementar el "truco". Es decir gritar truco, retruco y vale 4. Con la posibilidad de que la respuesta sea un simple, quiero o no quiero o el siguiente desafío.

No se alcanzó a implementar los casos de uso envido y flor, pero basados en la experiencia del truco, se cree que no sería difícil hacerlo. Para el caso del envido en particular porque se considera prácticamente análogo al de truco ya implementado. El caso de la flor es diferente pues según el estado de la mano hay que mostrar o no las cartas del contrario, para que verifique la flor, además que la casuística propia de la flor compone varios casos de uso diferentes.

Para concluir se pasó a probar la aplicación en dispositivos reales, para esto se contaba con un Nokia 5200, Samsung Omnia 2 y un Sony Ericsson Vivaz. En el Nokia no se pudo instalar la aplicación, mientras que en los otros dos, funcionó a la perfección. Claro está quitando lo ya mencionado en cuanto a que la resolución en ambos dispositivos es tal que las cartas quedan muy pequeñas. Un hecho a resaltar es que la aplicación fue desarrollada en emuladores, de dispositivos con botones físicos, e instalada en dispositivos "Touch Screen", funcionando correctamente en ambos, esto se debe a que LWUIT cumplió con lo prometido.

9. CONCLUSIONES Y TRABAJO A FUTURO

Como conclusión primera sobre el trabajo realizado, se puede indicar que JavaME es una plataforma relativamente buena para desarrollo de aplicaciones para dispositivos móviles. Como puntos positivos el desarrollar en Java, lo cual aplana mucho la curva de aprendizaje. Por otra parte teniendo en claro los conceptos de MIDLET, MIDP, CLDC se puede empezar a desarrollar una aplicación en minutos. Además si se le suma los JSR opcionales y los frameworks desarrollados sobre estos se tiene una plataforma muy completa. La cual simplifica mucho la utilización de todos los recursos de los dispositivos. Esto sin tener casi que tomar en cuenta cual es el fabricante, pues las interfaces abstraen lo suficiente como para preocupase por ello. Por otro lado se tienen las herramientas disponibles para el desarrollo. El WTK provee los emuladores, que son fundamentales para el testeo de la aplicación en las primeras etapas. Además de otras utilidades muy interesantes como el generador de clientes WS. A su vez el plugin EclipseME, a pesar de estar un poco obsoleto es lo suficientemente estable para aportar al desarrollo y no molestar con bugs y otros problemas.

El punto positivo de que está basado en Java, también tiene su lado negativo, pues da la falsa sensación de que se tiene todo el soporte de la plataforma Java, lo cual es una falacia, muchas de las librerías estándar de Java no están soportadas, y algunas solo están soportadas parcialmente, esto induce a errores que lamentablemente solo se detectan en tiempo de ejecución. Otro espejismo es creer que la aplicación desarrollada podrá correr en cualquier dispositivo cualquiera sea el fabricante. Esto no es así porque muchas de las interesantes librerías son opcionales y el fabricante puede decidir si incluirlas o no a su solo criterio. Por tanto una elección cuidadosa de que tecnologías se van a utilizar en la aplicación es necesaria para no tener sorpresas a posteriori.

En cuanto a la aplicación propiamente dicha, ya se ha dicho que está incompleta, por lo tanto se requieren muchas horas más de trabajo para dejarla en un esta aceptable para soportar el juego Truco en su totalidad. Por ejemplo una mejora posible seria utilizando un debugger contra las KVM de los celulares de prueba verificar porque se demora hasta 15 seg. para establecer una conexión bluetooth para evaluar como desde el lado de la aplicación se puede mejorar.

Otra mejora posible es la Ofuscación del Jar. Además de asegurar el código [20], la ofuscación brinda la posibilidad de mejorar la performance de la aplicación y de bajar el peso del jar que la contiene [21].

10. REFERENCIAS

[1] JavaME - Java ME at a Glance
http://www.oracle.com/technetwork/java/javame/overview/index.html
Fecha última visita sábado 4 de diciembre del 2010

[2] Truco - Truco (juego de naipes)
http://es.wikipedia.org/wiki/Truco_%28juego_de_naipes%29
Fecha última visita sábado 4 de diciembre del 2010

[3] Java 2D - Java 2D
http://es.wikipedia.org/wiki/Java_2D

Fecha última visita sábado 4 de diciembre del 2010
http://www.programacion.com/articulo/graficos_con_java_2d_111
Fecha última visita sábado 7 de diciembre del 2010

[4] JSR- Java Community Process
http://es.wikipedia.org/wiki/Java_Community_Process
Fecha última visita sábado 4 de diciembre del 2010

[5] Bluetooth - Bluetooth
http://es.wikipedia.org/wiki/Bluetooth
Fecha última visita sábado 4 de diciembre del 2010

[6] WS - Servicio web
http://es.wikipedia.org/wiki/Servicio_web
Fecha última visita sábado 4 de diciembre del 2010

[7] XML - Extensible Markup Language
http://es.wikipedia.org/wiki/Extensible_Markup_Language
Fecha última visita sábado 4 de diciembre del 2010

[8] CLDC - La arquitectura J2ME
http://grasia.fdi.ucm.es/j2me/_J2METech/index.html
Fecha última visita sábado 4 de diciembre del 2010

[9] MIDP - Introducción a MIDP 2.0
http://leo.ugr.es/J2ME/MIDP/intro.htm
Fecha última visita sábado 4 de diciembre del 2010

[10] Midlet - Midlet
http://es.wikipedia.org/wiki/Midlet
Fecha última visita sábado 4 de diciembre del 2010

[11] RMS - J2ME record management store
http://www.ibm.com/developerworks/library/wi-rms/
Fecha última visita sábado 4 de diciembre del 2010

[12] Layout - Layout manager
http://en.wikipedia.org/wiki/Layout_manager
Fecha última visita sábado 4 de diciembre del 2010

[13] LWUIT
Java ME Technology - LWUIT
http://www.oracle.com/technetwork/java/javame/tech/lwuit-141954.html
Fecha última visita sábado 4 de diciembre del 2010

Lightweight User Interface Toolkit
http://en.wikipedia.org/wiki/Lightweight_User_Interface_Toolkit
Fecha última visita sábado 4 de diciembre del 2010

[14] Sprites - Sprite (computer graphics)
http://en.wikipedia.org/wiki/Sprite_%28computer_graphics%29
Fecha última visita sábado 4 de diciembre del 2010

[15] OTA
Over-the-Air Provisioning with the J2ME Wireless Toolkit
http://developers.sun.com/mobility/midp/ttips/wtkota/
Fecha última visita sábado 4 de diciembre del 2010

Introduction to OTA Application Provisioning
http://developers.sun.com/mobility/midp/articles/ota/
Fecha última visita sábado 4 de diciembre del 2010

[16] Push Registry - The MIDP 2.0 Push Registry
http://developers.sun.com/mobility/midp/articles/pushreg/
Fecha última visita sábado 4 de diciembre del 2010

[17] JAD - JAD (file format)
http://en.wikipedia.org/wiki/JAD_%28file_format%29
Fecha última visita sábado 4 de diciembre del 2010

[18] JAR - Java Archive
http://es.wikipedia.org/wiki/Java_Archive
Fecha última visita sábado 4 de diciembre del 2010

[19] Marge - marge Project home
https://marge.dev.java.net/
Fecha última visita sábado 4 de diciembre del 2010

[20] Ofuscación - Ofuscación de código
http://www.iec.csic.es/criptonomicon/java/ofuscacion.html
Fecha última visita sábado 4 de diciembre del 2010

[21] Performance a través de la Ofuscación - Java Performance Tuning
http://www.javaperformancetuning.com/tips/j2me.shtml
Fecha última visita sábado 4 de diciembre del 2010