# ESCUELA POLITÉCNICA NACIONAL

## FACULTAD DE INGENIERÍA DE SISTEMAS

# DESARROLLO DE UNA BIBLIOTECA DE CLASES PARA MANEJO DE LA INTERFAZ GRÁFICA DE APLICACIONES MÓVILES BASADAS EN LA PLATAFORMA JAVA MICRO EDITION

**PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE COMPUTACIÓN**

**OQUENDO ROMERO CARLOS ANDRÉS**
andres18_04@yahoo.com

**DIRECTORA: ING. SANDRA SÁNCHEZ, MSC**
sandra.sanchez@epn.edu.ec

**QUITO, OCTUBRE 2011**

## DECLARACIÓN

Yo, Carlos Andrés Oquendo Romero, declaro bajo juramento que el trabajo aquí descrito es de mi autoría, que no ha sido previamente presentado para ningún grado o calificación profesional y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo a la Escuela Politécnica Nacional, según lo establecido en la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad constitucional vigente.

Carlos Andrés Oquendo Romero

## CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Carlos Andrés Oquendo Romero bajo mi supervisión.

Ing. Sandra Sánchez

**DIRECTORA DEL PROYECTO**

## AGRADECIMIENTOS

*A mis padres, Carlos Augusto y Yolanda, quienes con su trabajo, enseñanzas y consejos me han dado las bases para ser una persona de bien y de provecho.*

*A mis hermanos, Daniel y Anita, compañeros insuperables, con quienes la vida diaria siempre será mejor y más llevadera.*

*A mis tíos, especialmente a Edgar, Tere y Jorge, y a mis primos, en particular a Mauricio, por el apoyo que a su manera me entregaron para seguir adelante con este reto tan importante.*

*A mis compañeros de clase, con quienes he compartido la aventura de estudiar en las aulas de la Escuela Politécnica Nacional, y a mis profesores, por las enseñanzas impartidas.*

*A todos mis amigos y amigas, a quienes aprecio inmensamente.*

## DEDICATORIA

*A mis papás, Carlos Augusto y Yolanda, y a mis hermanos, Daniel y Anita, que siempre me brindan su soporte y cariño de manera incondicional.*

*A la memoria de la abuelita Juanita, que permanentemente nos cuida desde el cielo.*

# CONTENIDO

# LISTA DE TABLAS

# LISTA DE FIGURAS

# PRESENTACIÓN

En la actualidad, los teléfonos celulares son un bien a los que accede una gran parte de la población. Según cifras publicadas por la Superintendencia de Telecomunicaciones del Ecuador, la penetración de la telefonía móvil alcanza un 105% en el año 2011 (AmericaEconomia, 2011), y se estimó que para el año 2010 la penetración a nivel mundial de la telefonía móvil alcanzaría un 76% (mobiThinking, 2011). A medida que la tecnología avanza, estos aparatos han dejado de ser únicamente un mecanismo de comunicación y se han convertido en herramientas de productividad, y por ende incorporan nuevas funciones, brindan más prestaciones y permiten a los usuarios acceder a diversos servicios.

La mayoría de los fabricantes de dispositivos incorporan la plataforma Java Micro Edition en sus productos, especialmente de gamas media y baja, permitiendo a los desarrolladores crear aplicaciones que funcionen sobre estos equipos. A pesar de disponer de una plataforma estandarizada, la experiencia del usuario se ve afectada debido a la forma en la que una misma aplicación se presenta e interactúa con el usuario en uno u otro dispositivo.

# RESUMEN

El Proyecto de Titulación se centra en el desarrollo de una biblioteca de clases que permita el manejo de las interfaces gráficas de usuario de aplicaciones basadas en la plataforma Java ME, orientada específicamente a permitir que los desarrolladores elaboren aplicaciones móviles que tengan una apariencia actualizada y ofrezcan una experiencia de usuario consistente y estandarizada en cualquier dispositivo en el que se ejecuten.

En el primer capítulo se ofrece una visión general acerca de la plataforma Java Micro Edition, sus características y limitaciones, y se conoce acerca de la forma en que Java Micro Edition gestiona las interfaces gráficas de usuario de forma predeterminada. También se hace una revisión de la metodología de desarrollo de software OpenUP y se repasa las consideraciones que se deben hacer para desarrollar interfaces de usuario usables y funcionales, tomando en cuenta las características generales de los dispositivos móviles.

En el capítulo dos se describe el proceso de desarrollo de la biblioteca de clases siguiendo las prácticas establecidas por la metodología de desarrollo de software Open UP y aplicándolas en el caso práctico materia de este proyecto de titulación.

El tercer capítulo se centra en la construcción de un prototipo demostrativo que utiliza la biblioteca de clases desarrollada según lo descrito en el capítulo dos, así como en las pruebas que se ejecutan para comprobar el correcto funcionamiento de la biblioteca de clases en múltiples dispositivos con características y prestaciones diferentes.

Para terminar, en el capítulo cuatro, se incluye un compendio de las conclusiones obtenidas a partir del trabajo desarrollado y se expone una serie de recomendaciones basadas en la experiencia adquirida.

# CAPÍTULO 1. PLANTEAMIENTO DEL PROBLEMA

## 1.1. PLATAFORMA JAVA MICRO EDITION

La plataforma Java Micro Edition es una colección de tecnologías y especificaciones diseñadas concretamente para su uso en dispositivos móviles y sistemas embebidos.

Estas tecnologías se pueden combinar para construir un entorno de tiempo de ejecución Java completo, optimizado para ajustarse a los requisitos de un mercado o de equipos específicos, como aparatos industriales, electrodomésticos, y dispositivos personales. De esta última categoría, sobresalen por su naturaleza los teléfonos celulares.

### 1.1.1. EVOLUCIÓN DE LA PLATAFORMA

Sun Microsystems diseñó Java Micro Edition con la intención de reemplazar una tecnología anterior similar denominada PersonalJava.

El desarrollo de JME se realizó y se liberó como la especificación *JCP-68*. A partir de esta especificación se han creado diferentes versiones de Java Micro Edition, cada una detallada en un JSR diferente.

En Diciembre de 2006, Sun Microsystems lanzó el código fuente de JME bajo la licencia *GNU General Public License* con el nombre de phoneME. (Wikimedia Foundation, 2010)

### 1.1.2. CARACTERÍSTICAS DE JAVA MICRO EDITION

Sun Microsystems creo la tecnología Java Micro Edition con el objetivo de simplificar el desarrollo de aplicaciones para dispositivos pequeños y manejar adecuadamente sus restricciones.

Las condiciones básicas que se definieron para Java ME eran, principalmente, ajustarse a un entorno de trabajo reducido y posibilitar la creación de aplicaciones Java que se ejecuten en dispositivos con memoria limitada, pantalla pequeña y capacidad energética restringida.

La arquitectura de Java Micro Edition se encuentra compuesta por tres elementos fundamentales: configuraciones, perfiles y paquetes opcionales.

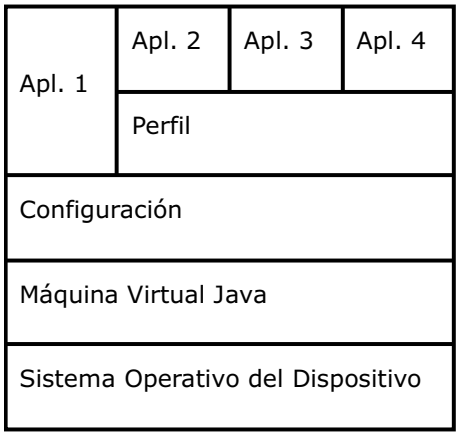



**Figura 1.1 Arquitectura General de Java Micro Edition**
Referencia: Wells, Martin J. J2ME Game Programming, 2005, P. 16
Traducido y adaptado por: Carlos Andrés Oquendo

Las *configuraciones* especifican una Máquina Virtual de Java y un conjunto limitado de interfaces de programación (API) para un amplio rango de dispositivos, similares en memoria y poder de procesamiento. En esencia, es la capa que minimiza la plataforma Java, removiendo algunos componentes, como partes del lenguaje, requisitos de hardware y bibliotecas adicionales.

Como parte del desarrollo de Java Micro Edition, Sun desarrolló dos configuraciones principales: CDC (Connected Device Configuration y CLDC (Connected, Limited Device Configuration)

Las configuraciones permiten que los desarrolladores cuenten con un conjunto estándar de funcionalidades básicas para cualquier dispositivo

Los *perfiles* son un conjunto de APIs especializadas, adicionales a las provistas por las configuraciones, que permiten manejar y estandarizar algunas funcionalidades que los fabricantes de dispositivos móviles pueden incluir en sus productos.

Los perfiles evitan que los desarrolladores deban utilizar APIs creadas específicamente para cada dispositivo, y les permiten desarrollar aplicaciones para una familia específica de dispositivos.

Los *paquetes opcionales* son un conjunto de APIs orientados hacia una tecnología específica, que los fabricantes pueden o no incluir en sus dispositivos.

### 1.1.3. PERFILES Y CONFIGURACIONES PRINCIPALES

Connected Device Configuration – CDC

Esta configuración se enfoca en dispositivos grandes, como televisores digitales, consolas de video, aparatos de televisión por cable, sistemas de navegación vehicular, PDAs de altas prestaciones, etc.

Un *dispositivo conectado* incluye usualmente un CPU de 32 bits, con al menos 2 MB de memoria RAM y unos 2.5 MB de memoria ROM disponible para el entorno de aplicación Java.

Las implementaciones de referencia para los perfiles de CDC se basan en Linux ejecutándose sobre un PC Intel compatible. (Wikimedia Foundation, 2010)

CDC especifica que los dispositivos que la implementen deben soportar una JVM completa de alto rendimiento, conocida como CVM o *Compact Virtual Machine*.

CDC contiene tres perfiles principales: *Foundation Profile, Personal Basis Profile* y *Personal Profile.*

Connected, Limited Device Configuration – CLDC

Es la configuración diseñada para ser implementada en dispositivos pequeños, como localizadores, teléfonos celulares, algunos tipos de teléfonos inteligentes, o PDAs sencillas. Define un estándar, utilizado por los fabricantes para construir e implementar un entorno de tiempo de ejecución Java. Los desarrolladores que siguen el estándar, acceden a una plataforma confiable donde ejecutar sus aplicaciones.

La implementación de referencia de CLDC se basa en una JVM pequeña y restringida, conocida como KVM o *Kilobyte Virtual Machine*.

CLDC se encarga de regular algunos aspectos del desarrollo Java, como:

- *Características de los dispositivos:* CLDC define las características mínimas que un dispositivo debe poseer, como al menos 192 KB de memoria RAM para la plataforma Java, 160 KB de memoria ROM para la máquina virtual y las librerías de CLDC, un procesador de 16 o 32 bits

corriendo al menos a 16 MHz, alguna forma de conectividad intermitente, inalámbrica y con bajo ancho de banda, y alguna fuente de energía, usualmente baterías.

- *Modelo de seguridad:* Se enfoca en dos aspectos fundamentales:
    - Seguridad de la máquina virtual: Protege al dispositivo de posibles daños generados por el código del usuario. Evita la ejecución de instrucciones inválidas o la creación de escenarios en los cuales la memoria externa al entorno Java se corrompa. Incluye la verificación del *bytecode* de las aplicaciones.
    - Seguridad de la aplicación: Protege el uso de los recursos del dispositivo. La plataforma Java ME incorpora un modelo de seguridad simplificado basado en un *sandbox,* donde se ejecuta el código de las aplicaciones Java ME, como se puede observar en la Figura 1.2.

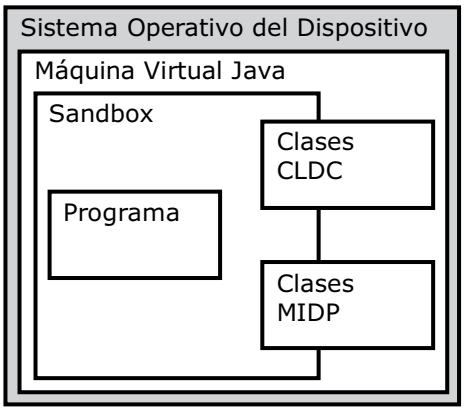



**Figura 1.2 Modelo de Seguridad de Aplicaciones Java ME**
Referencia: Wells, Martin J. J2ME Game Programming, 2005, P. 21
Traducido por: Carlos Andrés Oquendo

- *Administración de las aplicaciones:* El dispositivo debe incorporar cualquier funcionalidad que permita al usuario revisar, ejecutar o eliminar las aplicaciones instaladas. CLDC no especifica la forma que debe tener el administrador de aplicaciones.
- *Diferencias del lenguaje:* Como Java ME es un subconjunto de Java SE, algunas de sus funcionalidades son limitadas o inexistentes. Por ejemplo, no se dispone de aritmética de punto flotante (en CLDC 1.0), la finalización de los objetos es automática, no se incluyen algunas clases de manejo de errores de `java.lang.Error`, etc.

- *Diferencias con la JVM:* La KVM de CLDC no dispone de algunas características de la JVM genérica, como por ejemplo referencias débiles, reflexión, grupos de threads, threads de *background*, *Java Native Interface (JNI)* y cargadores de clases personalizados.
- *Bibliotecas de clases incluidas:* Se incluye un subconjunto de las clases existentes en Java SE, que se clasifican bajo el paquete `java.*` y clases propias de Java ME, contenidas en el paquete `javax.*`. No se incluyen clases relacionadas con el manejo de interfaz gráfica, como aquellas de los paquetes `javax.awt.*` y `javax.swing.*` ni las relacionadas con la conectividad, pertenecientes al paquete `java.net.*`.

Mobile Information Device Profile - MIDP

MIDP es el perfil más común y popular debido a que se convirtió en el estándar de facto para muchos juegos y aplicaciones.

Específicamente, MIDP extiende las funcionalidades ofrecidas por CLDC y proporciona funcionalidades específicas para los dispositivos conocidos como MID (*Mobile Information Device*).

Desde el punto de vista del entorno de hardware, MIDP, en su versión 2.0, establece las siguientes características.

- Pantalla con resolución mínima de 96x54 pixeles, con 1 bit de color y una relación de aspecto del pixel de casi 1 a 1.
- Medios de entrada como un teclado numérico (*keypad* de teléfono), un teclado alfanumérico (tipo QWERTY), o pantalla táctil.
- Memoria: Por lo menos 256 KB de memoria ROM para los componentes de MIDP, 8 KB de memoria no volátil para los datos persistentes generados por las aplicaciones, 128 KB de memoria RAM para usarse en tiempo de ejecución.
- Red: Conectividad inalámbrica de doble vía, usualmente intermitente, con un ancho de banda limitado.

El sistema operativo de los MID varía significativamente en funcionalidad y poder. Por lo tanto, MIDP establece algunas capacidades mínimas que los sistemas operativos deben cumplir:

- Memoria: Acceso a alguna forma de memoria no volátil.
- Red: Suficientes operaciones para el trabajo en red de los componentes de la API de MIDP.
- Gráficos: Capacidad de mostrar alguna forma de gráficos de mapa de bits.
- Medios de entrada y salida: Mecanismos para capturar la entrada de los usuarios y proveer retroalimentación.
- Kernel: Un kernel de sistema operativo funcional básico, capaz de manejar interrupciones, excepciones y con alguna forma de calendarizar procesos.

La implementación de MIDP provee algunas bibliotecas de clases que permiten a los desarrolladores acceder a:

- La interfaz de usuario del dispositivo,
- El subsistema de sonido del equipo,
- Medios de almacenamiento persistentes o *record store*,
- Funciones de red, que implementan el *framework* de conectividad definido en CLDC,
- Contadores y temporizadores,
- Funcionalidades de gestión de las aplicaciones.

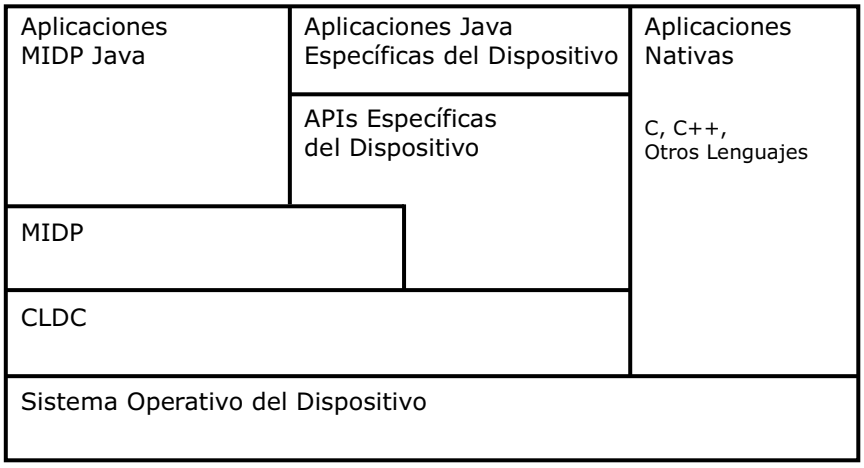



**Figura 1.3 Arquitectura de un sistema basado en MIDP**
Referencia: Knudsen J., Li, S. Beginning J2ME: From Novice To Professional, 2005, P. 8
Traducido por: Carlos Andrés Oquendo

### 1.1.4. LIMITACIONES DE LA PLATAFORMA

El desarrollo de aplicaciones móviles para la plataforma Java Micro Edition se ve afectado por las siguientes limitaciones generales de los dispositivos.

- ***Tamaño de la pantalla**:* El área disponible de la pantalla usualmente es reducida, comparado con el tamaño de las pantallas de computador.
- ***Conectividad de Red**:* Los dispositivos usualmente incluyen servicios de conectividad intermitente y limitada, en capacidad de transferencia de datos y en ancho de banda.
- ***Restricciones de memoria**:* La mayoría de los dispositivos incluyen relativamente poca memoria disponible para uso exclusivo del usuario, sin tomar en cuenta las tarjetas de expansión extraíbles. Usualmente la memoria del dispositivo es compartida para todas las funciones del teléfono y no puede utilizarse completamente para instalar aplicaciones.

Al considerar la plataforma de programación disponible para los dispositivos, se tiene que hacer las siguientes precisiones:

- La plataforma Java ME encuentra basada en la versión 1.3 del lenguaje Java, que tiene muchas restricciones en comparación con las versiones actuales de Java.
- Debido al modelo de seguridad implementado, Java ME no puede acceder a ciertas funciones nativas de los teléfonos.
- Las aplicaciones no pueden acceder sin autorización del usuario a funciones específicas, como conectividad, datos del usuario, sistema de archivos, etc., a menos que se encuentren firmadas digitalmente.
- Cada fabricante especifica la manera de implementar la funcionalidad de interfaz de usuario de alto nivel que pueden utilizar las aplicaciones.
- En algunos casos, los fabricantes no implementan o incluyen los paquetes opcionales que podrían ser requeridos para desarrollar exitosamente una aplicación móvil.

### 1.1.5. MODELOS DE INTERFAZ GRÁFICA

Las interfaces de programación que permiten a las aplicaciones acceder a la interfaz gráfica de los dispositivos que implementan la plataforma Java Micro

Edition no forman parte de la configuración CLDC, sino que son proporcionadas por el perfil MIDP.

Las clases para el manejo de la interfaz gráfica que MIDP proporciona no se basan en las tecnologías existentes para Java SE, como AWT (Abstract Window Toolkit) o SWING, debido a algunas restricciones funcionales propias de los dispositivos móviles, como el tamaño reducido de la pantalla, la resolución o la profundidad de colores, o los tipos de medios de entrada disponibles.

MIDP proporciona a los desarrolladores interfaces de programación de alto y bajo nivel, en el paquete `javax.microedition.lcdui`, que permiten administrar la interfaz gráfica de las aplicaciones basadas en este perfil.

El API de alto nivel, que se encuentra implementado en las diferentes subclases de `javax.microedition.lcdui.Screen`, se centra en permitir y facilitar el desarrollo de aplicaciones portables entre las diferentes marcas y modelos de dispositivos. Para lograr la portabilidad, se emplea un alto nivel de abstracción y, en consecuencia, el desarrollador obtiene muy poco control de la apariencia visual de la aplicación.

El sistema operativo del dispositivo se encarga de adaptar la interfaz de usuario de la aplicación al hardware específico del dispositivo y a su estilo y diseño gráfico nativo. Como resultado, la interfaz de usuario generada para la aplicación será completamente diferente, en dispositivos de diferentes marcas y modelos.

En contraste, el API de bajo nivel provee muy poca abstracción. Este API se diseñó para las aplicaciones que requieren acceder a los eventos de entrada de bajo nivel o que necesitan ubicar elementos con mayor precisión, permitiendo a los desarrolladores controlar completamente lo que se va a dibujar en la pantalla del dispositivo.

Este API se encuentra implementado en las clases `Canvas` y `Graphics` presentes dentro del paquete `javax.microedition.lcdui`. Cuando se utiliza estas clases no se garantiza la portabilidad de la aplicación, ya que el API provee algunos mecanismos para acceder a funcionalidades específicas de los dispositivos.

## 1.2. METODOLOGÍA OPEN UP

OpenUP es una variante sencilla y extensible del Proceso Unificado, que aplica un enfoque iterativo e incremental al ciclo de vida estructurado de un proyecto de desarrollo. OpenUP contempla una filosofía ágil y pragmática que se enfoca en aprovechar la naturaleza colaborativa del desarrollo de software.

El esfuerzo personal se organiza en micro incrementos, que representan unidades de trabajo cortas, que permiten que el proyecto avance a un ritmo continuo y medible, típicamente en horas o pocos días. El proceso promueve la colaboración intensiva de un equipo de trabajo comprometido y organizado a medida que se desarrolla el sistema. Cada micro incremento proporciona un periodo de retroalimentación muy corto que permite tomar decisiones de adaptación dentro de cada iteración.

OpenUP divide al proyecto en iteraciones planeadas con duraciones medidas casi siempre en semanas. Las iteraciones enfocan al equipo de trabajo en entregar incrementos con valor, de una manera predecible, de acuerdo a un plan de iteraciones definido. El resultado final es un producto que puede ser mostrado.

Los equipos de trabajo de OpenUP se organizan por su cuenta para cumplir los objetivos de la iteración y se comprometen a entregar los resultados, definiendo y tomando tareas detalladas de una lista de elementos de trabajo. OpenUP aplica un ciclo de vida de la iteración que estructura la forma de aplicar cada uno de los micro incrementos, y de esta manera, entregar productos estables y cohesivos que permitan que el sistema progrese de manera incremental para lograr los objetivos de la iteración.

El ciclo de vida de un proyecto OpenUP se divide en cuatro fases principales: *Inicio*, *Elaboración*, *Construcción* y *Transición*. Durante todo el transcurso del proyecto, existen puntos de visibilidad y de decisión para los interesados y miembros del equipo. De esta manera, se puede hacer un seguimiento efectivo y tomar decisiones en tiempos apropiados.

Un plan del proyecto define el ciclo de vida, que tiene como resultado final una aplicación terminada.

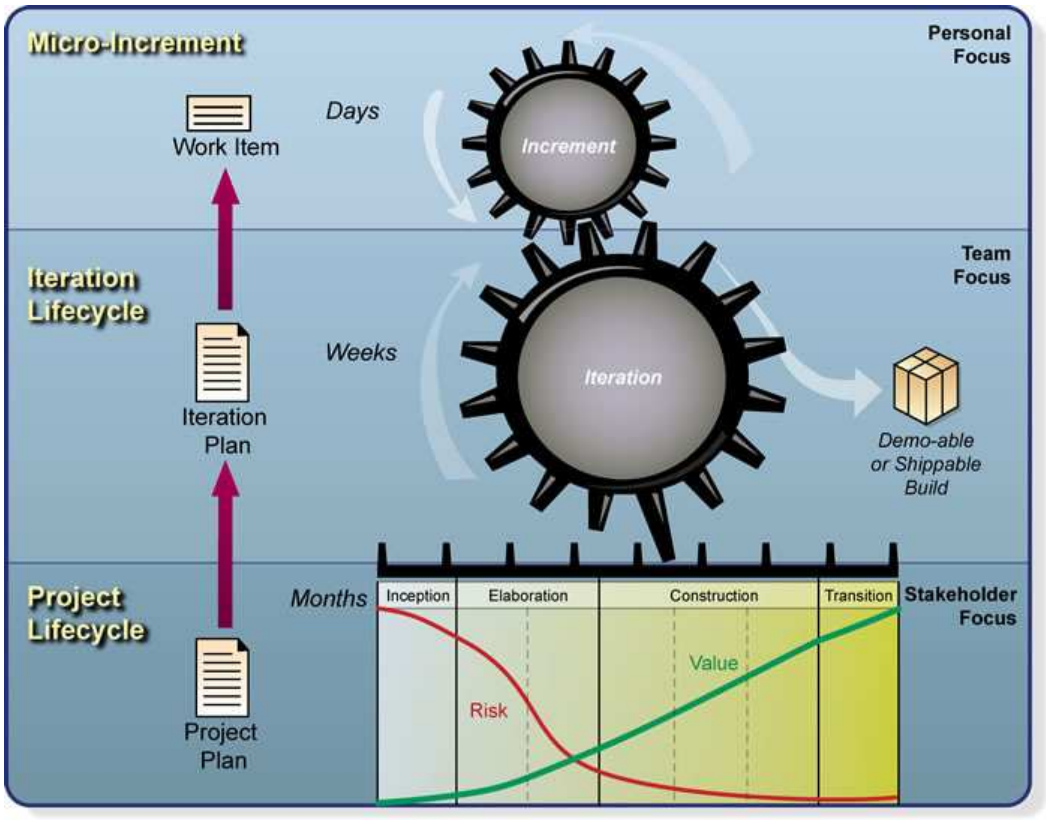



**Figura 1.4 Estructura de OpenUP**
Referencia: OpenUP Online Documentation
http://epf.eclipse.org/wikis/openup

## 1.3. CONSIDERACIONES DE DISEÑO DE INTERFACES

El medio por el cual los usuarios acceden e interactúan con un equipo de computación y sus aplicaciones de software se denomina interfaz de usuario. Existe una gran cantidad de posibles interfaces que los usuarios pueden utilizar para comunicarse con los diferentes sistemas de computación.

En el mundo de los dispositivos móviles, las interfaces de usuario comunes de la computación general de escritorio, que incluyen mecanismos de entrada y salida como el teclado, el ratón y el monitor, no son los más adecuados. En su lugar aparecen los teclados compactos (de tipo QWERTY, numéricos o alfanuméricos), las pantallas táctiles, los sistemas de alerta auditiva, etc.

Uno de los tipos de interfaz de usuario más común es la Interfaz Gráfica de Usuario, que en los dispositivos móviles, representa el medio más directo para visualizar las aplicaciones e interactuar con el dispositivo.

Al momento de diseñar interfaces gráficas de usuario adecuadas para cualquier aplicación, se debe considerar principalmente los factores humanos, entre los cuales se destacan:

- El *look and feel* de la aplicación.
- Los gustos de los usuarios.
- La facilidad de aprendizaje de la interfaz.
- La eficiencia que pueden lograr los usuarios al usar la interfaz.
- Cuestiones de salud al usar la interfaz de usuario.

Cuando se desarrolla interfaces gráficas de usuario enfocadas en los dispositivos móviles, se debe considerar además los siguientes aspectos:

- ***Ciclos cortos de transacciones:*** Los usuarios realizan tareas rápidas y sencillas. Evitan realizar tareas donde deban ingresar muchas cantidades de datos o que se tomen largos ciclos transaccionales.
- ***Alto desempeño de los dispositivos:*** Los usuarios esperan un alto desempeño de los dispositivos móviles y tiempos de respuesta bajos. Por ejemplo, es inadmisible que un dispositivo tome varios minutos para encenderse, arrancar y estar listo para su uso.
- ***Pérdida de foco:*** Los usuarios de dispositivos móviles no se enfocan en una tarea de computación. Usualmente los usuarios se encuentran haciendo varias cosas a la vez.

De esta manera, se puede establecer algunos aspectos importantes para el desarrollo de interfaces gráficas de usuario para los dispositivos móviles.

- ***Tiempo de respuesta del sistema y tiempo de acceso del sistema:*** Éstos son un problema común en la computación. Es inaceptable que una aplicación produzca esperas largas, ya que distraen al usuario de usar la aplicación, sus características o el dispositivo como tal.
- ***Interfaces gráficas limpias y eficientes:*** No es adecuado que las interfaces gráficas de usuario sean cargadas o apiñadas. La interfaz gráfica de un sistema móvil debe tener solamente los elementos necesarios para brindar al usuario la información y funcionalidades que requiere y nada más.

- ***Consistencia entre las interfaces de usuario:*** Es muy importante que los usuarios tengan la misma experiencia de usuario a medida que usan los diferentes módulos de una aplicación.
- ***Tener en cuenta las limitaciones y capacidades humanas:*** Considerar que no todas las personas reaccionan igual ante los estímulos recibidos por los diferentes sentidos. No se debe saturar los sentidos del usuario.
- ***Priorizar hábilmente los elementos de la interfaz de usuario:*** La aplicación móvil debe poder mostrar u ocultar elementos de la interfaz en función de condiciones externas, como la conexión de red, el estado de la batería, el nivel del volumen, etc.

## 1.4. CONSIDERACIONES DE USABILIDAD

A continuación se definen algunos aspectos que permiten que las aplicaciones, en general, sean más usables:

- ***Intuición:*** Las interfaces gráficas de usuario deben ser intuitivas, de tal manera que cuando el usuario use la interfaz por primera vez, pueda navegarla y examinarla sin mayor problema.
- ***Consistencia:*** Los componentes de la interfaz de usuario deben ser consistentes entre si y, de ser posible, con su entorno operativo.
- ***Facilidad de aprendizaje:*** El usuario debe ser capaz de aprender a usar la interfaz con pocas veces de uso y volver a usarla sin utilizar manuales.
- ***Ayuda no intrusiva:*** La interfaz gráfica y la aplicación deben proveer ayuda y consejos que no afecten el uso eficiente de la aplicación.
- ***Acomodarse a los usuarios expertos:*** Una buena interfaz permite a los usuarios expertos trabajar más rápido, usualmente usando atajos, que se conocen a medida que se usa más el sistema.
- ***Confiable:*** La interfaz gráfica debe ser predecible, confiable y entendible con facilidad. Debe permitir que el usuario adivine la reacción de la interfaz ante una acción determinada.
- ***Robustez:*** Una buena interfaz debe ser rápida, recuperarse correctamente de los errores del usuario, mostrar retroalimentación frente a las acciones del usuario, y permitirle saber si debe esperar por respuestas.

## 1.5. ANÁLISIS DE SOLUCIONES EXISTENTES

En el mercado existen algunas soluciones que permiten administrar la interfaz gráfica de las aplicaciones móviles basadas en la plataforma Java Micro Edition.

Se destacan las siguientes:

### 1.5.1. LIGHT WEIGHT USER INTERFACE TOOLKIT - LWUIT

LWUIT es una biblioteca para el manejo de la interfaz gráfica en aplicaciones Java ME orientada a dispositivos de uso masivo. Ofrece una interfaz de programación inspirada en SWING. Su propósito es evitar que los desarrolladores deban escribir código orientado a un dispositivo específico o a diferentes tamaños de pantalla.

El soporte de LWUIT se extiende a dispositivos que incluyan como plataforma a MIDP 2.0 y CLDC 1.1. Es una tecnología abierta, con el código fuente y los binarios accesibles libremente para uso comercial o individual bajo la licencia GPL con Excepción de Classpath.

Actualmente, Oracle promueve el uso de LWUIT como componente oficial de para la plataforma Java Micro Edition.


**Figura 1.5 Ejemplo de la Interfaz Gráfica LWUIT**
Referencia: Java ME Technology – LWUIT
http://java.sun.com/javame/technology/lwuit/

### 1.5.2. J2ME POLISH - LUSH

J2ME Polish es un conjunto de herramientas para desarrolladores de aplicaciones móviles. El conjunto completo ofrece funcionalidades de manejo de UI, persistencia, comunicaciones, base de datos y *porting*.

LUSH es el componente de J2ME Polish que permite a los desarrolladores gestionar las interfaces de usuario de sus aplicaciones. LUSH proporciona un conjunto de controles para construir las interfaces gráficas y además proporciona la posibilidad de diseñar y personalizar estas interfaces de usuario utilizando técnicas similares a las hojas de estilo en cascada.


**Figura 1.6 Ejemplo de la Interfaz Gráfica J2ME Polish – LUSH**
Referencia: J2ME Polish: Documentation
http://www.j2mepolish.org/cms/leftsection/documentation/design/visual-guide.html

### 1.5.3. MEWT

Es un conjunto de controles de usuario que permite crear interfaces gráficas para aplicaciones basadas en Java Micro Edition.

Los controles proporcionados por MEWT se adaptan automáticamente a los diferentes tamaños de las pantallas minimizando el esfuerzo de probar y depurar una aplicación en varios tipos de dispositivos.

Incorpora controles adicionales como tablas, árboles, botones gráficos, y soporta la creación y aplicación de temas.


**Figura 1.7 Ejemplo de la Interfaz Gráfica MEWT**
Referencia: Mewt – J2ME Widget Toolkit
http://mewt.sourceforge.net/theme-screenshots.html

### 1.5.4. J4ME UI

Java For ME (J4ME) UI es una librería que permite construir aplicaciones Java Micro Edition. Incluye funcionalidades para construir interfaces de usuario, registro de eventos y acceso a los dispositivos GPS.

En el lado de la interfaz gráfica, J4ME se centra en permitir la construcción de aplicaciones con apariencia profesional e intuitiva. Se utiliza en lugar del marco de trabajo LCDUI, presente de manera predeterminada en la plataforma Java ME.

Ofrece a los desarrolladores la posibilidad de crear interfaces gráficas que se pueden personalizar usando temas, con operaciones de botones consistentes entre teléfonos móviles y promoviendo el desarrollo orientado a objetos.

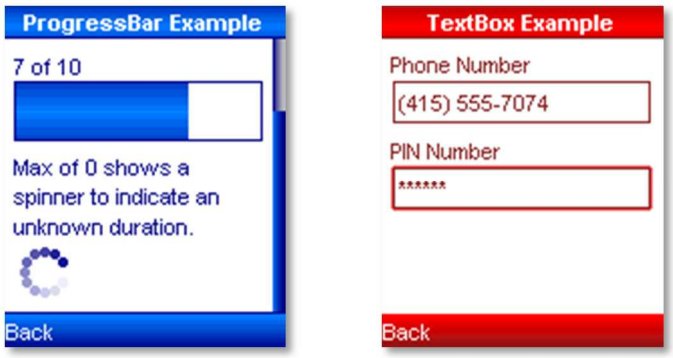



**Figura 1.8 Ejemplo de la Interfaz Gráfica J4ME**
Referencia: UI - j4me - The J4ME UI
http://code.google.com/p/j4me/wiki/UI

### 1.5.5. KUI

kUI es un marco de trabajo de interfaz gráfica orientado a reemplazar algunas de las clases LCDUI, con otras basadas a partir del API de bajo nivel Canvas, principalmente para reducir la complejidad y para ahorrar espacio de almacenamiento y memoria. Estos objetivos se obtienen concentrando la funcionalidad de algunas clases pertenecientes a LCDUI en un pequeño conjunto de clases kUI.

Según el fabricante, kUI puede ser estilizado para ajustarse a los requerimientos personales, evita problemas generados por los comandos *Close* implícitos, no requiere ejecución de la fase de pre procesamiento y ofrece un comportamiento y experiencia consistente entre diferentes dispositivos, simplificando el proceso de desarrollo y documentación.

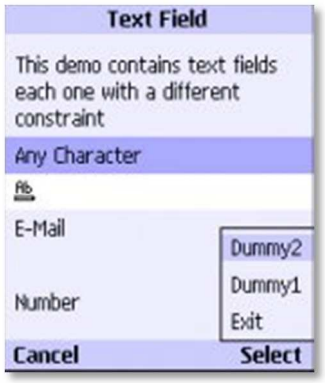



**Figura 1.9 Ejemplo de la Interfaz de Usuario kUI**
Referencia: Sourceforge.net: kUI: Screenshots
http://sourceforge.net/project/screenshots.php?group_id=191343

### 1.5.6. RESUMEN DE CARACTERÍSTICAS DE PRODUCTOS DE TERCEROS

La Tabla 1.1 muestra un resumen de algunas características referenciales de cada una de las soluciones mencionadas anteriormente.

| Producto | Licencia | Versión | MIDP | CLDC | Temas | Fecha de Versión | Productor |
|---|---|---|---|---|---|---|---|
| ***LWUIT*** | Apache 2.0 | 1.3 | 2.0 | 1.1 | Si | Mar 2010 | Oracle Corporation |
| ***J2ME Polish*** | GNU GPL, Propietaria | 2.1.4 | 2.0 | 1.1 | Si | Mar 2010 | Enough Software |
| ***mewt*** | Apache 2.0, GNU GPL | 1.0.4 | 2.0 | 1.1 | Si | Jun 2008 | Web Hiker |
| ***J4ME UI*** | Apache 2.0 | 1.0.3 | 2.0 | 1.1 | Si | May 2008 | Score Out |
| ***kUI*** | GNU GPL, Propietaria | N/A | 2.0 | 1.1 | Si | Aprox. 2008 | kObjects |

**Tabla 1.1. Comparativo de las Soluciones de Terceros existentes**
Elaborado por: Carlos Andrés Oquendo

A simple vista podemos observar que, de las soluciones presentadas, apenas dos se encuentran en desarrollo activo y, cuentan con soporte de los productores. El resto de soluciones, no ha tenido mayor actividad durante el último año.

Uno de los problemas principales que se ha detectado durante la revisión realizada a los sitios web de los productos es la falta de documentación del producto y técnica. En algunos casos, los productores ofrecen solo la documentación de programación (javadoc) y FAQs. En otros casos, se proveen también mini aplicaciones que ejemplifican el uso de los productos.

Todos los *frameworks* proporcionan funcionalidades de manejo de la interfaz gráfica, sin embargo J2ME Polish y J4ME incluyen también funcionalidades relacionadas con otras áreas y tecnologías, como el registro de mensajes y eventos, manejo de dispositivos de posicionamiento GPS, comunicaciones, etc.

Se puede notar que las diferentes soluciones trabajan proveyendo alternativas al uso de los componentes existentes dentro de la biblioteca LCDUI.

En general, las interfaces gráficas generadas por las soluciones se ven algo desactualizadas y, en algunos casos, no se ajustan a ciertas tendencias de diseño gráfico actual, que se puede ver a continuación en las Figuras 1.10, 1.11 y 1.12.

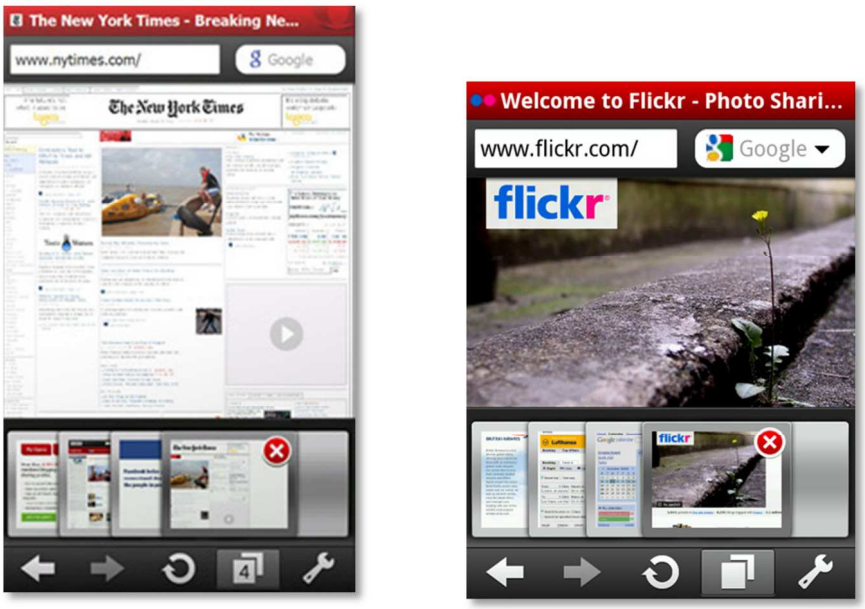



**Figura 1.10 Interfaz Gráfica de Opera Mobile y Opera Mini**
Fabricante: Opera Software ASA
Fuentes: http://www.opera.com/press/releases/2010/03/16,
http://www.opera.com/press/releases/2009/11/18

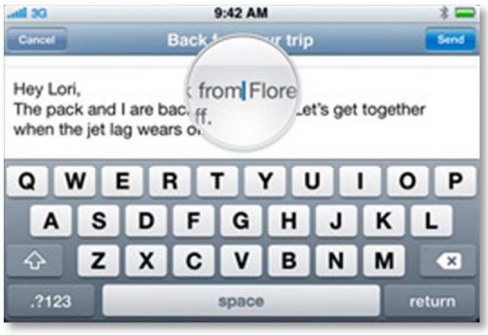



**Figura 1.11 Interfaz de Usuario del Apple iPhone**
Fabricante: Apple Inc.
Fuente: http://www.apple.com/la/iphone


**Figura 1.12 Interfaz de Usuario de la Aplicación F1 2010**
Fabricante: Formula One Administration
Fuente: http://www.formula1.com/mobile_services/live_timing.html

## 1.6. CARACTERÍSTICAS DE LA SOLUCIÓN PROPUESTA

La solución propuesta se orientará a brindar un marco de trabajo para el desarrollo de interfaces de usuario con las siguientes características y funcionalidades.

- Proveer un mecanismo adecuado de manejo de las interfaces de usuario en tiempo de ejecución.
- Permitir la personalización de la interfaz de usuario y sus controles a través del uso de *temas*.
- Proporcionar una colección de controles estándar acordes con el estilo visual de la interfaz de usuario, como botones, cajas de texto, etiquetas, selectores, etc.
- Implementar algunos controles de usuario nuevos, como barras de herramientas, botones, barras de navegación, cuadros de diálogo, etc.
- Entregar una apariencia y funcionalidad lo más genérica y consistente posible en diferentes modelos y marcas de teléfonos celulares.

# CAPÍTULO 2. DESARROLLO DE LA BIBLIOTECA DE CLASES

En el desarrollo de este capítulo se efectuará una revisión detallada de la metodología de desarrollo de software Open UP, que guiará los esfuerzos para la construcción exitosa de la solución propuesta, que ha sido denominada como *User Interface Library for MIDP – UIL 4 MIDP*

Como parte de este proceso de revisión, se encuentra previsto generar sobre la marcha los artefactos propuestos por la metodología. Todos los artefactos elaborados se expondrán como anexos y, en caso de ser necesario, algunos serán incluidos dentro del cuerpo de este capítulo.

## 2.1. FASE DE INICIO

La fase de inicio del proyecto permite realizar una primera definición de la solución que se tiene previsto construir.

La metodología de desarrollo establece los siguientes objetivos principales:

- Entender lo que se va a construir.
- Identificar la funcionalidad clave del sistema.
- Determinar una posible solución.
- Comprender los costos, riesgos y tiempos vinculados al proyecto.

El cumplimiento de éstos objetivos está asociado con la ejecución de las actividades que se describen a continuación.

### 2.1.1. INICIAR EL PROYECTO

Esta actividad se realiza por una sola vez al comienzo de la primera iteración. Con esta actividad se intenta establecer la visión para el proyecto y su planificación, con un alto nivel de granularidad.

Al finalizar esta actividad se obtienen como artefactos un documento de visión, un glosario, y un plan para el proyecto.

#### 2.1.1.1. DESARROLLO DE LA ACTIVIDAD

La primera acción que se debe llevar a cabo es la identificación de todos los interesados, directos o indirectos, en la construcción de la solución.

Aplicación Práctica: UIL 4 MIDP

Se ha identificado a los siguientes actores como interesados en la construcción de la biblioteca de clases:

- ***Desarrolladores externos*****,** que construyen aplicaciones Java ME, y que podrían, eventualmente, usar la solución como elemento base de sus aplicaciones,
- ***Usuarios***, que utilizan las aplicaciones diseñadas y elaboradas por los desarrolladores externos,
- ***Analista del sistema***, quien se encarga de determinar los requerimientos para la solución,
- ***Arquitecto de software***, que lidera todo el proceso del desarrollo del sistema, y,
- ***Desarrollador interno***, que tiene a su cargo la construcción de la solución.

De todos los interesados enumerados, solamente los *Desarrolladores externos* y los *Usuarios* son relevantes para obtener información que permita realizar la definición de la solución.

Una vez que se ha determinado y descrito a los interesados, se procede a definir el problema a solucionar, tomando en cuenta los temas considerados como *inconvenientes* por parte de los interesados que han sido calificados como relevantes.

Aplicación Práctica: UIL 4 MIDP

En la Tabla 2.1 se encuentra un resumen de los problemas identificados por el grupo de interesados relevantes.

| Interesado | Problemas |
|---|---|
| ***Desarrollador Externo*** | El diseño de interfaces gráficas personalizadas, usando las funcionalidades genéricas es muy complicado y requiere de bastante tiempo. |
| | Algunas de las soluciones de terceros no tienen soporte ni documentación, son desactualizadas y no tienen un desarrollo activo. |
| | Los marcos de trabajo genéricos de la plataforma para crear interfaces gráficas (LCDUI) no permiten mayor control a los desarrolladores. |
| | Las interfaces gráficas de la misma aplicación, basada en marcos de trabajo genéricos, lucen y tienen otro comportamiento en diferentes dispositivos. |
| ***Usuario*** | Las aplicaciones en general no son visualmente atractivas, a excepción de los juegos. |

**Tabla 2.1. Problemas reconocidos por los interesados de la solución**
Elaborado por: Carlos Andrés Oquendo

De forma paralela, se realiza la formulación del planteamiento del problema, y se lo plasma en el documento de visión.

El siguiente paso consiste en recolectar los requerimientos que los interesados tienen para la solución.

Aplicación Práctica: UIL 4 MIDP

Los requerimientos recolectados de parte de los interesados han sido incluidos en la Tabla 2.2.

| Interesado | Requerimientos |
|---|---|
| ***Desarrollador Externo*** | Flexibilidad y rapidez para el diseño de interfaces gráficas para los *MIDlets*. |
| | Posibilidad de crear aplicaciones con interfaces gráficas más atractivas y modernas, incluyendo componentes actualizados. |
| | Compatibilidad de las aplicaciones con múltiples dispositivos móviles que utilizan la plataforma Java ME. |
| | Provisión y existencia de documentos de ayuda y soporte adecuados. |
| ***Usuario*** | Aplicaciones móviles más fáciles e intuitivas para usar. |
| | Respuesta ágil de las aplicaciones. |

**Tabla 2.2. Requerimientos de los Interesados externos**
Elaborado por: Carlos Andrés Oquendo

En este punto, al disponer de una base de conocimiento general de los problemas y los requerimientos expuestos por los interesados, se procede a establecer y convenir el conjunto de características que la solución debería soportar.

Para cada una de las características puntualizadas se incluirá una breve descripción y una serie de atributos complementarios.

Aplicación Práctica: UIL 4 MIDP

En la Tabla 2.3 se aprecian las características convenidas para la solución, con la información adicional relevante.

| Característica | Descripción | Atributos |
|---|---|---|
| ***Componentes visuales actualizados*** | Conjunto de componentes visuales que tienen una funcionalidad específica, como botones, cuadros de texto, etc. | ▪ Apariencia atractiva al usuario.<br>▪ Manejo de eventos del usuario.<br>▪ Funcionamiento consistente entre dispositivos.<br>▪ Facilidades de personalización. |
| ***Mecanismos de organización de los componentes visuales*** | Funcionalidades que permiten gestionar la forma en que los componentes visuales se disponen y aparecen en la pantalla del dispositivo. | ▪ Organización automática de los componentes.<br>▪ Cambio automático de tamaños y posiciones.<br>▪ Control de la navegación entre pantallas y vistas. |
| ***Soporte a múltiples dispositivos*** | La solución deberá permitir la construcción de aplicaciones que funcionen sin modificación en múltiples dispositivos móviles. | ▪ Utilización de funcionalidad del marco de trabajo genérico.<br>▪ Detección del tamaño de la pantalla.<br>▪ Mínima utilización de recursos del sistema. |
| ***Documentación de programación adecuada.*** | Provisión de documentos que ayuden a la programación, adicionales a la documentación estándar. | ▪ Ayuda de programación en línea.<br>▪ Documentos con ejemplos, guías y explicaciones. |

**Tabla 2.3. Características propuestas para la solución**
Elaborado por: Carlos Andrés Oquendo

Al concluir esta tarea, se dispone de suficiente información para generar los artefactos planteados al inicio de esta sección.

### 2.1.1.2. ARTEFACTOS

Documento de Visión

Determina el enfoque que tienen los interesados acerca de la solución técnica que debe ser desarrollada, en función de necesidades clave y de requerimientos técnicos

Captura la solución en función de peticiones y de restricciones técnicas o no, que han sido manifestadas por los interesados, y brinda una vista a alto nivel del razonamiento, contexto y fondo que permiten establecer requerimientos más detallados.

# User Interface Library for MIDP - UIL 4 MIDP Visión

## 1. Introducción

### 1.1. Propósito

El propósito de este documento es recoger y definir a alto nivel las necesidades, requisitos y características para el sistema User Interface Library for MIDP (UIL 4 MIDP). La manera en que el sistema cubrirá estas especificaciones se detalla en los casos de uso y especificaciones adicionales.

### 1.2. Alcance

Este documento aplica a la solución User Interface Library for MIDP (UIL 4 MIDP), que permite a los desarrolladores de aplicaciones móviles basadas en la plataforma Java Micro Edition crear y administrar la interfaz gráfica de sus aplicaciones.

## 2. Posicionamiento

### 2.1. Planteamiento del Problema

| | |
|---|---|
| El problema de | La falta de un marco de trabajo moderno para el desarrollo de interfaces gráficas para las aplicaciones Java Micro Edition, de tal manera que se vean y funcionen de manera consistente en diferentes dispositivos |
| afecta a | Los desarrolladores de aplicaciones Java Micro Edition<br>Los usuarios de las aplicaciones Java Micro Edition |
| el impacto de lo cual es | No disponer de aplicaciones con interfaces gráficas llamativas y modernas |
| una solución exitosa sería | Aquella que facilite y permita la creación de interfaces gráficas de usuario modernas para las aplicaciones Java Micro Edition, que permita que las aplicaciones tengan una apariencia moderna, ágil y consistente y que funcione de la misma manera en diferentes dispositivos. |

### 2.2. Planteamiento de la Posición del Producto

| | |
|---|---|
| Para | Los desarrolladores de aplicaciones basadas en la plataforma Java Micro Edition |
| Quienes | Desarrollan aplicaciones móviles que incluyen principalmente interfaces gráficas de usuario |
| El producto UIL 4 MIDP | Es una biblioteca de clases |
| Que | Permite la creación de interfaces gráficas de usuario para las aplicaciones Java Micro Edition |
| A diferencia de | Las soluciones comerciales y libres existentes como<br>▪ J4ME ▪ MEWT<br>▪ LWUIT ▪ Otras |
| Este producto | Proporciona un marco de trabajo para la creación de interfaces de usuario modernas para las aplicaciones Java Micro Edition, de tal manera que su apariencia y funcionalidad permanezca consistente entre diferentes dispositivos. |

# 3. Descripción de los Interesados

## 3.1. Resumen de los Interesados

| Nombre | Descripción | Responsabilidades |
|---|---|---|
| Desarrollador Externo | Consumidor directo de la biblioteca de clases. | Construye aplicaciones móviles utilizando como base la plataforma Java Micro Edition con interfaces gráficas basadas en el producto propuesto. |
| Usuarios | Usuarios de las aplicaciones móviles creadas a partir de la biblioteca de clases. | Utilizan las aplicaciones cuya interfaz gráfica se crea utilizando la biblioteca de clases, validan su funcionamiento y apariencia, y proveen retroalimentación. |
| Analista del Sistema | Determina los requerimientos de la solución. | Establece los requerimientos de la solución delineando y delimitando la funcionalidad del sistema. |
| Arquitecto de Software | Lidera el desarrollo del sistema. | Define la arquitectura de la solución, que incluye las decisiones técnicas claves, que restringen el diseño global y la implementación del proyecto. |
| Desarrollador Interno | Construye la solución | Se encarga de implementar la solución siguiendo los lineamientos establecidos por los demás involucrados. Genera productos finales para uso externo. |

## 3.2. Entorno del Usuario

Los desarrolladores externos pueden utilizar las clases implementadas en la biblioteca UIL 4 MIDP para construir la interfaz gráfica de usuario de sus aplicaciones Java Micro Edition. La funcionalidad interna de las aplicaciones no se ve afectada por la implementación de la interfaz gráfica.

# 4. Descripción del Producto

## 4.1. Necesidades y Características

| Necesidad | Prioridad | Características | Planificación de Lanzamiento |
|---|---|---|---|
| Componentes visuales actualizados | Alta | ▪ Apariencia atractiva al usuario.<br>▪ Manejo de eventos del usuario.<br>▪ Funcionamiento consistente entre dispositivos.<br>▪ Facilidades de personalización. | Versión 1 |
| Mecanismos de organización de los componentes visuales | Alta | ▪ Organización automática de los componentes.<br>▪ Cambio automático de tamaños y posiciones.<br>▪ Control de la navegación entre pantallas y vistas. | Versión 1 |
| Soporte a múltiples dispositivos | Alta | ▪ Utilización de funcionalidad del marco de trabajo genérico.<br>▪ Detección del tamaño de la pantalla.<br>▪ Mínima utilización de recursos del sistema. | Versión 1 |
| Documentación de programación adecuada | Media | ▪ Ayuda de programación en línea.<br>▪ Documentos con ejemplos, guías y explicaciones. | Versión 1 |

**5. Otros Requisitos del Producto**

| Requisito | Prioridad | Planificación de Lanzamiento |
|---|---|---|
| Soporte a dispositivos basados en CLDC 1.1 y MIDP 2.0 por lo menos. | Alta | Versión 1 |
| Alto rendimiento de la solución | Alta | Versión 1 |
| Mínima ocupación de espacio de almacenamiento. | Media | Versión 1 |

Glosario

Mantiene los términos y definiciones importantes para el proyecto. Este artefacto permite clarificar ideas y ayuda a brindar consistencia y entendimiento común para todos los involucrados, durante todo el ejercicio de las actividades planificadas.

Siempre se debe contar con un glosario o un documento similar, en la forma que sea más conveniente para el desarrollo del proyecto.

Cuando el tamaño del proyecto es muy pequeño o los términos utilizados son muy familiares para los interesados, es posible obviar la creación y el uso de este artefacto.

Para la solución, como este artefacto evoluciona durante el proyecto, no se incluye en esta sección.

Plan del Proyecto

Recoge toda la información necesaria que permite gestionar el proyecto desde un nivel y con un enfoque estratégico.

Proporciona un documento central donde se encuentra y mantiene toda la información necesaria respecto a la forma en que se llevará a cabo la ejecución del proyecto.

La metodología especifica que en cualquier tipo de proyecto se debe disponer de este artefacto, inclusive para aquellos proyectos de tamaño pequeño o aquellos cuya complejidad es reducida o limitada.

# User Interface Library for MIDP - UIL 4 MIDP
# Plan del Proyecto

## 1. Introducción

Este plan describe el desarrollo de la solución denominada User Interface Library for MIDP (UIL 4 MIDP).

## 2. Organización del Proyecto

El trabajo necesario para desarrollar el proyecto se realizará por Carlos Andrés Oquendo, quien será responsable de la generación de los productos, artefactos y entregables requeridos, haciéndose cargo de uno o más roles según sea el caso. La supervisión del avance del proyecto será responsabilidad de la Ing. Sandra Sánchez, que actúa como Directora del Proyecto de Titulación.

## 3. Prácticas y Medidas del Proyecto

El proyecto se desarrollará siguiendo un enfoque iterativo basado en micro incrementos, como se define en la metodología Open UP. De esta manera, se puede evaluar de manera más efectiva el avance del proyecto.

## 4. Hitos y Objetivos del Proyecto

Los objetivos principales del proyecto de desarrollo de la solución UIL 4 MIDP son:

| Fase | Iteración | Objetivos Primarios | Duración |
|---|---|---|---|
| Inicio | I1 | 1. Definir el proyecto a alto nivel<br>2. Determinar necesidades, características y requerimientos<br>3. Definir riesgos | 15 Días Hábiles |
| Elaboración | I2 | 1. Refinar requerimientos.<br>2. Desarrollar la arquitectura<br>3. Desarrollar prototipos y pruebas de concepto | 20 Días Hábiles |
| Construcción | I3 | 1. Refinar requerimientos.<br>2. Desarrollar incrementos.<br>3. Probar la solución. | 30 Días Hábiles |

## 5. Despliegue

El producto final se entregará en forma de código fuente y código objeto, compilados en forma de una biblioteca de clases, que podría ser utilizada directamente por los desarrolladores de MIDlets.

## 6. Lecciones aprendidas

- Se debe poner especial atención en los procesos de captura y refinamiento de requerimientos.
- Es necesario revisar el tiempo asignado a cada una de las fases del proyecto en forma meticulosa.

### 2.1.2. PLANEAR Y GESTIONAR LAS ITERACIONES

Esta actividad, que se lleva a cabo en forma permanente durante todo el ciclo de vida del proyecto, permite realizar la identificación, evaluación y mitigación de

riesgos y problemas potenciales con suficiente anticipación, establecer las metas y objetivos válidos para cada iteración y, ayudar a los interesados a cumplir las metas que han sido propuestas.

Como resultado final de la actividad, se genera el plan de la iteración, una lista de riesgos y una lista de elementos de trabajo.

#### 2.1.2.1. DESARROLLO DE LA ACTIVIDAD

Esta actividad inicia estableciendo objetivos más específicos, que deberán alcanzarse durante las iteraciones, en función de los objetivos de alto nivel que se determinaron en la definición y el alcance del proyecto.

Los objetivos deberán considerar las expectativas y funcionalidades requeridas por los interesados.

Aplicación Práctica: UIL 4 MIDP

Los objetivos más específicos identificados para las iteraciones del proyecto, se encuentran enumerados en la Tabla 2.4.

| No. | Objetivos específicos |
|---|---|
| 1 | Determinar la arquitectura para la biblioteca de clases. |
| 2 | Diseñar la apariencia visual que se debe implementar en la biblioteca de clases |
| 3 | Construir un conjunto de componentes de usuario que se adapten a la apariencia visual establecida. |
| 4 | Construir una serie de componentes auxiliares para el manejo de los componentes de usuario elaborados. |
| 5 | Asegurar permanentemente el funcionamiento de la solución en varios dispositivos. |
| 6 | Escribir la documentación de programación para la biblioteca de clases. |
| 7 | Escribir ayudas y guías como parte auxiliar de la documentación de programación. |

**Tabla 2.4. Objetivos para las iteraciones de la solución.**
Elaborado por: Carlos Andrés Oquendo

A continuación se debe establecer los elementos generales de trabajo para la solución, incluyendo datos adicionales como el esfuerzo y el tiempo estimado.

Toda esta información sirve como insumo para generar la lista completa de elementos de trabajo que contempla Open UP.

Aplicación Práctica: UIL 4 MIDP

Los elementos de trabajo que se han identificado en primera instancia para el proyecto se encuentran enumerados en la Tabla 2.5.

| Tarea | Tiempo Estimado | Prioridad |
|---|---|---|
| Determinar la arquitectura de la biblioteca de clases. | 16 horas | 1 |
| Establecer el conjunto de componentes a construir. | 8 horas | 2 |
| Diseñar la apariencia de la interfaz gráfica. | 32 horas | 1 |
| Construir y documentar un subconjunto básico de componentes. | 80 horas | 2 |
| Construir aplicaciones de prueba de los componentes básicos. | 24 horas | 2 |
| Construir y documentar los componentes restantes. | 96 horas | 3 |
| Construir aplicaciones de prueba con los componentes restantes | 32 horas | 3 |
| Escribir prototipos que demuestren funcionalidades específicas | 32 horas | 1 |

**Tabla 2.5. Listado preliminar de elementos generales de trabajo**
Elaborado por: Carlos Andrés Oquendo

A medida la ejecución de las actividades del proyecto avanza, aparecen nuevos temas que deben ser considerados, y que generan riesgos que podrían convertirse en problemas si no se identifican y neutralizan adecuadamente.

En este punto es conveniente revisar las decisiones tomadas y verificar la existencia de riesgos y problemas potenciales.

Aplicación Práctica: UIL 4 MIDP

En la Tabla 2.6 aparece el conjunto de riesgos identificados en la primera iteración.

| No. | Riesgo |
|---|---|
| 1 | Tiempo de construcción de los componentes mal estimado |
| 2 | Diseño defectuoso de la apariencia de la interfaz gráfica |
| 3 | Dificultad para implementar funcionalidades clave para la solución |
| 4 | Falta de dispositivos donde probar los prototipos y la solución |
| 5 | Cambios en las especificaciones de los marcos de trabajo genéricos. |
| 6 | Incompatibilidad con las plataformas implementadas por los diferentes fabricantes. |

**Tabla 2.6. Listado de riesgos generales del proyecto**
Elaborado por: Carlos Andrés Oquendo

La idea principal es que, al analizar los riesgos identificados, se puede completar y refinar la lista de elementos de trabajo con actividades que permitan mitigar, resolver o eliminar los riesgos.

Tanto la Lista de Riesgos como la Lista de Elementos de Trabajo se incluirán más adelante en el cuerpo de este documento.

Para concluir con esta tarea, se debe destacar que el trabajo de planificación de la iteración no finaliza con la generación de una única versión de los artefactos. A medida que avanza el proyecto se deberá evaluar las condiciones del momento así como los resultados obtenidos, para actualizar las listas de riesgos y de elementos de trabajo.

#### 2.1.2.2. ARTEFACTOS

Plan de la Iteración

Es un plan detallado que enuncia los objetivos, las asignaciones de trabajo y los criterios de evaluación para la iteración. En este artefacto se capturan los hitos de la iteración, puntos de sincronización con otros equipos y problemas que se deben resolver en la iteración, etc.

La metodología Open UP determina que no es necesario producir este artefacto cuando en el proyecto no van a realizar múltiples iteraciones, o cuando el alcance del proyecto puede ser administrado de manera informal en el interior del equipo de desarrollo.

El desarrollo de la solución UIL4MIDP se lleva a cabo por una sola persona, que es la que gestiona el alcance. Por estos motivos, se ha determinado que no se van a generar planes de iteración.

Lista de Riesgos

Enumera los riesgos conocidos y abiertos del proyecto, ordenados por magnitud descendente, con acciones de contingencia y mitigación específicas.

La metodología sugiere usar este artefacto siempre, a menos que se evidencie que el impacto de los riesgos identificados es mínimo.

## Lista de Riesgos: UIL 4 MIDP

| ID | Título | Descripción | Tipo | Impacto | Posibilidad | Magnitud | Estrategia de Mitigación |
|---|---|---|---|---|---|---|---|
| 1 | Dificultad para implementar funcionalidades clave para la solución. | La solución requiere la implementación de algunos algoritmos específicos, como el manejo de gradientes y colores, redimensionamiento de objetos, etc. | D | 4 | 65% | 2,6 | Crear prototipos puntuales que permitan implementar las funcionalidades requeridas. |
| 2 | Falta de dispositivos para probar los prototipos y la solución. | Los dispositivos considerados para la verificación de la solución no se encuentran disponibles temporal o permanentemente. | D | 5 | 50% | 2,5 | Procurar asegurar el préstamo u obtención de los dispositivos necesarios para probar la solución |
| 3 | Diseño defectuoso de la apariencia de la interfaz gráfica | La interfaz gráfica diseñada no tiene una apariencia atractiva y moderna. | D | 5 | 50% | 2,5 | Validar el diseño realizado con varios individuos para recoger sugerencias y mejoras. |
| 4 | Tiempo de construcción de los componentes mal estimado | El tiempo especificado para culminar la solución es demasiado corto, o se ve afectado por causas externas. | D | 4 | 50% | 2,0 | Procurar optimizar los tiempos existentes para la implementación de la solución. |
| 5 | Incompatibilidad con las plataformas implementadas por los diferentes fabricantes | La solución no es lo suficientemente genérica para funcionar sin cambios en las plataformas operativas consideradas. | D | 5 | 40% | 2,0 | Probar permanentemente la solución usando las herramientas provistas por los fabricantes durante toda la construcción. |
| 6 | Cambios en las especificaciones de los marcos de trabajo genéricos. | Oracle hace modificaciones profundas en la arquitectura de Java ME y en la implementación de referencia de la plataforma operativa. | I | 4 | 10% | 0,4 | Verificar que la solución se adapta a los cambios realizados, sin mayor impacto. |

**Lista de Riesgos: UIL 4 MIDP**
Elaborado por: Carlos Andrés Oquendo

Lista de Elementos de Trabajo

Registra las tareas que se deben realizar durante el proyecto. Para cada uno de los elementos de trabajo presentes en el listado se debe establecer una prioridad, estimar un tamaño y asignar un estado.

Los estados que los elementos de trabajo podrán tomar son: No Asignado, En Progreso, Completado.

Para la solución, se ha establecido una lista general de elementos de trabajo, que serán refinados y detallados a medida que el proyecto avance.

## Lista de Elementos de Trabajo: UIL 4 MIDP

| Nombre / Descripción | Prioridad | Tamaño Estimado (Puntos) | Estado |
|---|---|---|---|
| Determinar la arquitectura de la biblioteca de clases. | 1 | 2 | No Asignado |
| Establecer el conjunto de componentes a construir. | 2 | 2 | No Asignado |
| Diseñar la apariencia de la interfaz gráfica. | 1 | 6 | No Asignado |
| Validar el diseño de la interfaz gráfica | 2 | 4 | No Asignado |
| Escribir prototipos que demuestren funcionalidades específicas | 1 | 6 | No Asignado |
| Construir y documentar un subconjunto básico de componentes. | 2 | 10 | No Asignado |
| Construir aplicaciones de prueba de los componentes básicos. | 2 | 4 | No Asignado |
| Construir y documentar los componentes restantes. | 3 | 10 | No Asignado |
| Construir aplicaciones de prueba de los componentes restantes | 3 | 4 | No Asignado |

**Lista de Elementos de Trabajo: UIL 4 MIDP**
Elaborado por: Carlos Andrés Oquendo

Para apreciar de mejor manera la distribución de estas tareas, se ha elaborado el diagrama de Gantt, que se aprecia en la Figura 2.1, a continuación:

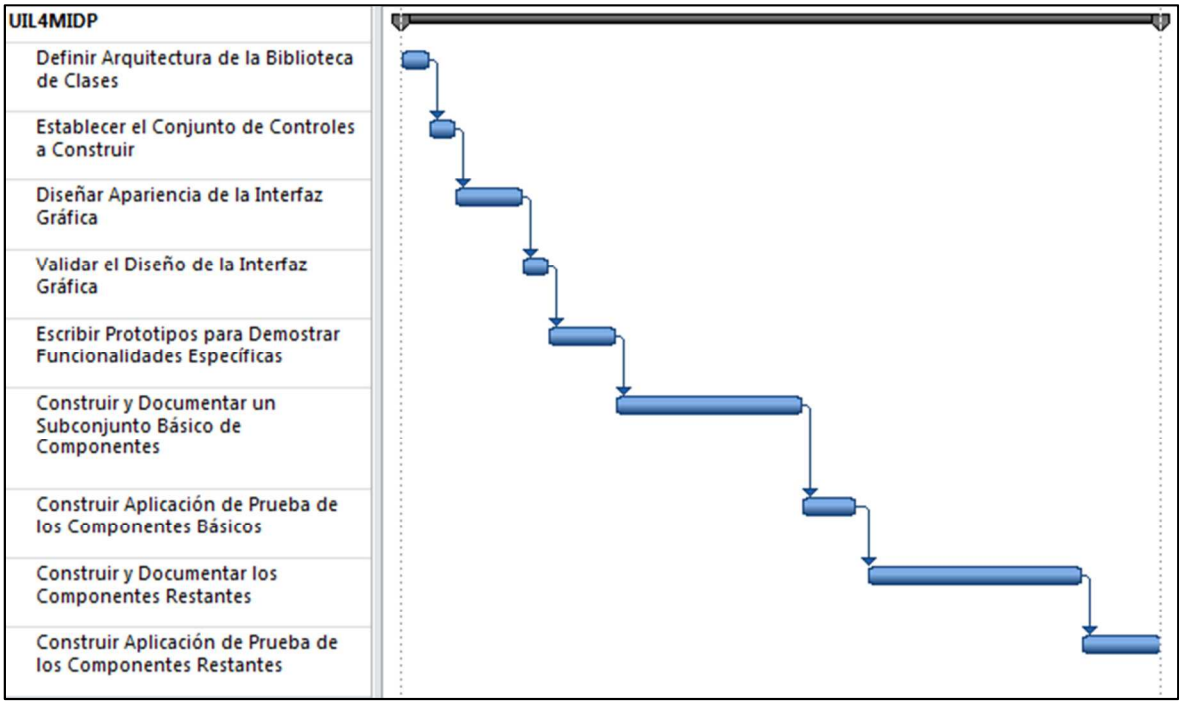



**Figura 2.1. Diagrama de Gantt de los Elementos de Trabajo Generales**
Elaborado por: Carlos Andrés Oquendo

El diagrama de Gantt siempre se mantendrá actualizado y reflejará las actividades detalladas en la Lista de Elementos de Trabajo.

### 2.1.3. IDENTIFICAR Y REFINAR LOS REQUERIMIENTOS

Permite recolectar, detallar, analizar y validar un subconjunto de requerimientos antes de proceder a la implementación.

En la fase de inicio, el enfoque de esta actividad se centra en delimitar el problema a resolver en función de las necesidades de los involucrados y capturando, a alto nivel, las características del sistema.

La metodología Open UP establece que los artefactos producto de esta actividad son un glosario más refinado y completo, un documento de especificaciones de requerimientos globales para la solución, casos de prueba, un modelo de casos de uso y las especificaciones de cada uno de los casos de uso.

#### 2.1.3.1. DESARROLLO DE LA ACTIVIDAD

Para iniciar con esta actividad, es necesario revisar y tener clara la información que se ha materializado en los artefactos generados durante las actividades anteriores, en especial, en el documento de visión.

A medida que se realiza la revisión, se debe profundizar en las peticiones y necesidades establecidas por los interesados, de tal manera que se pueda lograr un mejor entendimiento del dominio de la solución.

En forma paralela, se debe ir completando el glosario, con nuevos términos y definiciones que eventualmente podrían aparecer.

Aplicación Práctica: UIL 4 MIDP

Como parte de este proceso, se ha obtenido la siguiente tabla de requerimientos más detallados, tipificados como funcionales y no funcionales.

| Requerimiento | Tipo |
|---|---|
| Proveer un conjunto básico de componentes visuales. | Funcional |
| Incluir mecanismos que permitan la organización automática de los componentes visuales. | Funcional |

| | |
|---|---|
| Soporte a múltiples dispositivos | No funcional |
| Incluir mecanismos que faciliten la navegación y el desplazamiento entre las diferentes vistas de una aplicación. | Funcional |
| Los componentes visuales deben ser atractivos, tener una apariencia moderna, agradable y amigable. | No funcional |
| Se debe permitir la personalización de los componentes visuales. | Funcional |
| Los componentes visuales deben ajustarse automáticamente al tamaño de la pantalla. | Funcional |
| La solución debe utilizar el mínimo posible de recursos del sistema y tener un alto rendimiento. | No funcional |
| Se debe incluir documentación de programación como parte de la solución. | No funcional |
| Se debe proporcionar documentos con ejemplos, guías y explicaciones. | No funcional |
| El comportamiento de los componentes visuales debe ser predecible y consistente | No funcional |
| La solución debe trabajar en la plataforma estándar de Java ME (CLDC 1.1 y MIDP 2.0) | No funcional |
| No se debe requerir el uso de hardware especial con la solución. | No funcional |
| Los componentes de la solución deben tener la capacidad de reconocer eventos de la aplicación. | Funcional |

**Tabla 2.7. Requerimientos funcionales y no funcionales de la solución**
Elaborado por: Carlos Andrés Oquendo

El siguiente paso consiste en realizar la identificación de los actores y los casos de uso que aplican para la solución.

Aplicación Práctica: UIL 4 MIDP

Específicamente, para esta solución, se debe hacer las siguientes precisiones:

1. Ningún usuario (persona) puede utilizar o consumir directamente las funcionalidades de la solución. Esto sucede porque la solución se incluye siempre como un componente interno de una aplicación.
2. Las operaciones y las funcionalidades que la solución incorpora no representan acciones de valor para los usuarios (personas), es decir, no se enfocan en las operaciones propias de la solución, sino en las operaciones que realizan las aplicaciones que incorporan la solución.

De esta manera se puede determinar que las aplicaciones, desarrolladas en base

a la solución (MIDlets), son los actores directos. Tomando en consideración esta decisión se han identificado los siguientes casos de uso generales para la solución:

- Dibujar Vista
- Propagar Eventos de Teclado

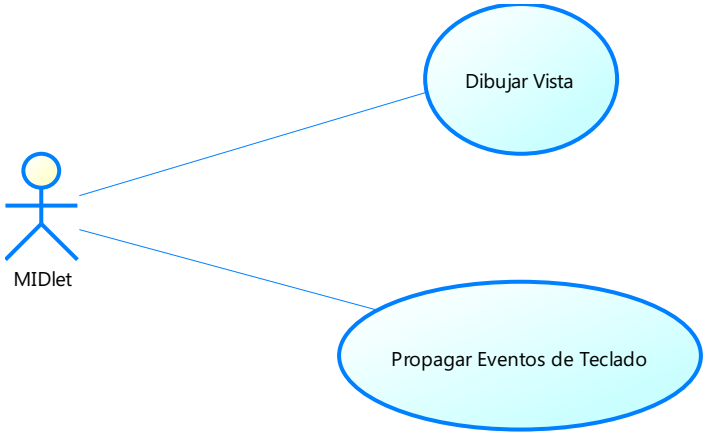



**Figura 2.2 Diagrama de Casos de Uso de UIL 4 MIDP**
Elaborado por: Carlos Andrés Oquendo

Finalmente, se podrán incorporar algunas definiciones correspondientes al licenciamiento de la solución.

Aplicación Práctica: UIL 4 MIDP

El licenciamiento de la solución UIL4MIDP se establecerá en base a un modelo *dual*, que permite publicar la solución bajo una licencia propietaria y una licencia de software libre.

Este tipo de licenciamiento dual se utiliza con frecuencia para diferenciar el enfoque de uso y distribución del software, y en consecuencia, permitir al usuario seleccionar una versión del producto en función de sus necesidades específicas, y además, permitir el financiamiento del desarrollo del producto.

Para uso de la solución UIL4MIDP se ha seleccionado la licencia de software libre Apache 2.0 (elaborada por la Apache Software Foundation). La licencia propietaria no se definirá todavía.

En este punto ya se puede empezar a construir los artefactos considerados por la metodología como resultados de esta actividad.

#### 2.1.3.2. ARTEFACTOS

Glosario

El glosario generado en las actividades descritas en la sección 2.1.1 de este documento debe ser revisado y actualizado. Como parte de este proceso, se puede refinar las definiciones y términos existentes e incorporar términos y descripciones adicionales.

Como este artefacto se encuentra en permanente actualización, no se lo incluye en esta sección del documento.

Especificación de Requerimientos Globales del Sistema

Este artefacto permite capturar y establecer, de manera general, los requisitos funcionales, los atributos cualitativos y las restricciones que impactan de manera global a la solución. Cuando se construye este artefacto, hay que asegurarse de representar las necesidades de todos los involucrados e interesados en el desarrollo del proyecto.

Eventualmente pueden aparecer algunos requerimientos que se sobreponen, no son muy específicos o no son lo suficientemente claros.

## User Interface Library for MIDP - UIL 4 MIDP
## Especificación de Requerimientos Globales del Sistema

### 1. Introducción

Este documento recoge los requerimientos globales para la solución denominada UIL 4 MIDP - User Interface Library for MIDP.

### 2. Requerimientos funcionales globales del sistema

- Proveer un conjunto básico de componentes visuales.
- Incluir mecanismos que permitan organizar de manera automática los componentes visuales en la pantalla.
- Incorporar mecanismos que faciliten la navegación y el desplazamiento entre las diferentes vistas de una aplicación.
- Introducir mecanismos que permitan la personalización de los componentes visuales.
- Establecer mecanismos que permitan a los controles ajustarse automáticamente al tamaño de la pantalla del dispositivo móvil.
- Los componentes de la solución deben tener la capacidad de reconocer eventos para la aplicación.

# 3. Características del sistema

## 3.1. Usabilidad

- Los componentes visuales deben ser atractivos, tener una apariencia moderna, agradable y amigable.
- El comportamiento de los componentes visuales debe ser predecible y consistente entre las diferentes vistas de una aplicación.
- Se debe incluir documentación de programación como parte de la solución.
- Se debe proporcionar documentos adicionales con ejemplos, guías y explicaciones.

## 3.2. Fiabilidad

- La solución debe ser robusta, de tal forma que no contribuya a generar problemas en las aplicaciones que la utilicen como componente base.

## 3.3. Rendimiento

La solución debe utilizar el mínimo posible de recursos del sistema y tener un alto rendimiento. Por esto, la solución no debe generar mayor impacto en el rendimiento de los programas que la incorporan, considerando los siguientes aspectos:

- Procurar utilizar la memoria estrictamente necesaria para realizar las operaciones internas.
- Evitar que la solución utilice recursos del sistema innecesariamente.
- Intentar optimizar la ejecución de los algoritmos que la solución utiliza.

## 3.4. Compatibilidad

- La solución deberá funcionar en los dispositivos móviles que implementen la plataforma estándar de Java Micro Edition, es decir, al menos la configuración CLDC 1.1 y el perfil MIDP 2.0.
- La solución debe poderse utilizar en múltiples dispositivos, de diferentes fabricantes.
- La solución no debe requerir el uso de hardware especializado.
- La solución debe ser ligera, es decir, debe utilizar el mínimo posible de recursos del sistema, y debe tener un tamaño reducido, que facilite su inclusión como parte de otras aplicaciones.

# 4. Interfaces del sistema

## 4.1. Interfaces de usuario

### *4.1.1. Aspecto*

- Los componentes visuales de la solución deben incluir una apariencia básica de manera predeterminada, que se encuentre acorde con las tendencias de diseño gráfico actual.

### *4.1.2. Diseño y navegabilidad*

- La solución debe incluir mecanismos que faciliten a los usuarios el desplazamiento entre las diferentes vistas de una aplicación.

### *4.1.3. Consistencia*

- Los componentes visuales provistos deberán comportarse de manera consistente y predecible, es decir, el comportamiento de las diferentes instancias de un componente, en diferentes vistas, debe ser el mismo, e independiente de las operaciones subyacentes.
- Las interfaces de programación incluidas deberán apegarse al uso de patrones de programación conocidos y convenientes para el mejor aprovechamiento de la solución, como por ejemplo, *singleton,* fábrica (*factory*), métodos de fábrica, constructores (*builder*)*, etc*.

### *4.1.4. Personalización*

- Se debe permitir la personalización de los componentes visuales. Esta personalización comprende el cambio de colores principales y secundarios, el cambio del tamaño del texto y el ajuste de las dimensiones de los componentes cuando se muestran en pantalla.

### 4.2. Interfaces a sistemas o dispositivos externos

*4.2.1. Interfaces de software*

- La solución tomará como punto de partida algunas de las clases básicas para la programación de interfaces de usuario provistas en la especificación MIDP 2.0

*4.2.2. Interfaces de hardware*

- La solución no depende de ningún componente de hardware especial para funcionar de manera nativa.

*4.2.3. Interfaces de comunicaciones*

- La solución no debe hacer uso de los mecanismos de comunicaciones provistos por los dispositivos.

## 5. Reglas de negocio

Las reglas de negocio se enfocan en representar o explicar los procesos de negocio que realizan las personas y que pueden ser automatizados. Como esta solución se puede utilizar como un componente interno de otras aplicaciones, no existen reglas de negocio que puedan ser aplicables directamente a la biblioteca de clases como tal.

## 6. Limitaciones del sistema

- La solución debe implementarse utilizando el lenguaje de programación Java y las funcionalidades disponibles en la plataforma Java Micro Edition, compuesta de al menos la configuración CLDC 1.1 y MIDP 2.0

## 7. Conformidad del sistema

### 7.1. Licenciamiento

La solución se distribuirá bajo un esquema de licenciamiento dual, es decir, bajo una licencia privativa diseñada específicamente para la solución y bajo una licencia de software libre.

La licencia privativa permitirá licenciar la solución para su uso como plataforma de aplicaciones cerradas o privativas, en aplicaciones cuyo esquema de licenciamiento sea incompatible con la licencia de software libre, en situaciones donde se requiera que se provea soporte a la solución, o la inclusión de funcionalidades especiales.

La licencia de software libre permitirá utilizar la solución en proyectos de cualquier índole, siempre y cuando se observen las disposiciones expuestas en ella. Cuando la solución se licencia en esta modalidad, no se proveerá soporte técnico.

Para la solución se ha decidido utilizar la Licencia Apache 2.0, desarrollada por la Apache Software Foundation. Entre sus características destacan:

- Es una licencia de software libre.
- No es una licencia de *copyleft*, es decir, no requiere que los trabajos derivados de la solución se redistribuyan bajo la misma licencia original.
- Requiere la preservación de las notas de *copyright* y de descargo de responsabilidades.
- Permite el uso del código fuente para el desarrollo de software propietario y/o software libre.

### 7.2. Notas legales, de derechos de autor, y otras

Como la solución es un proyecto de titulación, la Escuela Politécnica Nacional se reserva los derechos de autor y de propiedad intelecual de la solución. El autor de la solución, concede, sin derecho a reconocimiento económico por parte de la Escuela Politécnica Nacional, sus derechos de autor y de propiedad intelectual relativos a la solución UIL 4 MIDP una vez que el proyecto de titulación haya sido completado y aceptado por la Escuela Politécnica Nacional.

Una vez que la solución esté en poder de la Escuela Politécnica Nacional, si desea distribuirla, se podrá

cambiar el esquema de licenciamiento propuesto. Los interesados en licenciar la solución deberán contactarse con la Escuela Politécnica Nacional.

Al utilizar cualquiera de los dos esquemas de licenciamiento propuestos, la solución, conformada por el código fuente y compilado y su documentación asociada, se proveen tal cual, y no se ofrece garantía de ningún tipo, expresa o implícita, sobre la solución y tampoco el licenciatario se hace responsable de eventualidades causadas por el uso, o la falta de uso de la solución.

### 7.3. Estándares aplicables

El ciclo de vida del proyecto de software que permitirá obtener la solución, será gestionado usando la metodología Open UP.

Los diagramas de diseño se realizarán aplicando los estándares disponibles para UML.

La codificación de los componentes de la solución se realizará aplicando los estándares de programación adecuados para el lenguaje Java en su variante Micro Edition: MIDP 2.0 (Mobile Information Device Profile) y CLDC 1.0 (Connected, Limited Device Configuration).

## 8. Documentación del sistema

Se proveerá de la documentación técnica adecuada que permita hacer uso de la solución de forma natural. La documentación deberá escribirse usando un lenguaje y explicaciones que ayuden a los desarrolladores a aprovechar las funcionalidades de la solución.

En la solución se incluirá la documentación de las interfaces de programación y se la hará disponible en tiempo de diseño para facilidad de los desarrolladores.

Especificaciones de Casos de Uso y Modelo de Casos de Uso

Con estos artefactos se puede capturar, visualizar, establecer y explicar el comportamiento del sistema y su entorno para producir resultados de valor para los actores que interactúan con él.

Si no se cuenta con estos artefactos, la funcionalidad que el sistema debe soportar podría no ser clara, a menos que se utilicen formas alternativas de especificación de requerimientos.

El modelo de casos de uso correspondiente a la solución UIL4MIDP se muestra a continuación, utilizando la plantilla definida por la metodología Open UP para este artefacto.

# User Interface Library for MIDP - UIL 4 MIDP
# Modelo de Casos de Uso

## 1. Introducción

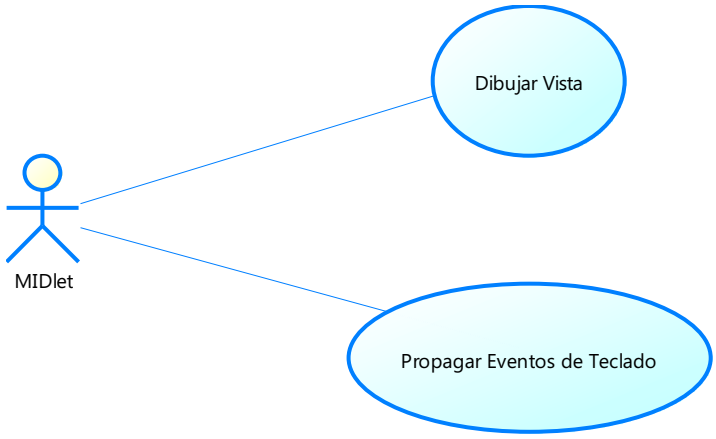
Este modelo de casos de uso permite identificar los actores primarios y los casos de uso que aplican para la solución UIL 4 MIDP.

## 2. Perspectiva General

UIL 4 MIDP es una biblioteca de clases que puede ser utilizada para crear las interfaces gráficas de usuario de aplicaciones basadas en la plataforma Java Micro Edition (MIDlet).

UIL 4 MIDP aparece como componente interno de estas aplicaciones.

## 3. Diagrama de Casos de Uso




## 4. Actores

### *4.1. MIDlet*

Este actor representa a una aplicación que se ejecuta sobre la plataforma Java Micro Edition (MIDP 2.0, CLDC 1.1) cuyas interfaces de usuario se construyen utilizando los componentes provistos por la solución UIL 4 MIDP.

## 5. Casos de Uso

### *5.1. Dibujar Vista*

Este caso de uso describe la forma en que el MIDlet dibuja una vista en la pantalla..

### *5.1. Propagar Eventos de Teclado*

Este caso de uso describe cómo el MIDlet propaga los eventos de teclado a los componentes de UIL 4 MIDP para su procesamiento.

# User Interface Library for MIDP - UIL 4 MIDP
# Caso de Uso: Dibujar Vista

## 1. Descripción

Este caso de uso describe la forma en que el MIDlet dibuja una vista en la pantalla.

## 2. Actores

### 2.1. MIDlet

## 3. Precondiciones

La aplicación utiliza las clases de UIL 4 MIDP para crear sus vistas y gestionar la interfaz gráfica de la aplicación.

Se ha asignado un objeto Window de UIL 4 MIDP a la instancia de Display del MIDlet.

## 4. Flujo de Eventos Básico

1. El caso de uso comienza cuando el MIDlet requiere mostrar una vista en pantalla.
2. El objeto Display llama al método de pintado perteneciente a Window.
3. El objeto Window delega al objeto de tipo Controller el pintado de la pantalla.
4. El objeto de tipo Controller, determina que vista debe ser mostrada, la selecciona y delega a esta el pintado de sus controles.
5. El objeto de tipo View recibe la llamada, y empieza a dibujar en pantalla sus controles privados.
6. Cuando el objeto View ya no tiene más controles privados que pintar, empieza a buscar los contenedores y controles de contenido que debe pintar. A cada uno les asigna posición y dimensiones y les delega la tarea de pintarse a sí mismos.
7. La vista se muestra en la pantalla.
8. El caso de uso termina.

## 5. Flujos Alternativos

N/A

## 6. Subflujos

N/A

## 7. Escenarios Clave

### 7.1. La vista no está compuesta por controles propios, contenedores u otros controles.

1. La vista se pinta vacía.

## 8. Post condiciones

### 8.1. Visualización exitosa de la vista.

La vista, con todos sus controles y contenedores se muestra en pantalla y se encuentra lista para responder a los eventos de teclado.

## 9. Requerimientos especiales

N/A

# User Interface Library for MIDP - UIL 4 MIDP
# Caso de Uso: Propagar Eventos de Teclado

## 1. Descripción

Este caso de uso describe cómo el MIDlet propaga los eventos de teclado a los componentes de UIL 4 MIDP para su procesamiento.

## 2. Actores

### 2.1. MIDlet

## 3. Precondiciones

La aplicación utiliza las clases de UIL 4 MIDP para crear sus vistas y gestionar la interfaz gráfica de la aplicación.

Se ha asignado un objeto Window de UIL 4 MIDP a la instancia de Display del MIDlet.

Existe un objeto Controller asignado a Window, y un objeto View activo en el objeto Controller.

## 4. Flujo de Eventos Básico

1. El caso de uso comienza cuando el MIDlet recibe un evento de teclado de parte del sistema operativo del dispositivo y de la plataforma Java ME.
2. El objeto Display del MIDlet es notificado, y este llama al método de manejo de eventos de teclado de Window.
3. El objeto Window delega al objeto de tipo Controller el procesamiento del evento de teclado recibido.
4. El objeto de tipo Controller, delega a la vista activa el procesamiento del evento de teclado.
5. El objeto de tipo View recibe la llamada, y determina quién (control o contenedor) debe procesar el evento de teclado. Cuando se ha determinado el destinatario del evento de teclado la vista lo propaga.
6. El caso de uso termina.

## 5. Flujos Alternativos

### 5.1. Vista no tiene control activo

Si en el paso 5, la vista determina que ningún componente visual se encuentra activo todavía, ella intentará procesarlo.

a) Si recibe eventos de teclas de desplazamiento, intentará activar un componente visual.

b) Si recibe otros eventos de tecla, no hará nada.

### 5.2. Vista tiene un contenedor como componente visual activo

Si en el paso 5, la vista determina que se encuentra activo un contenedor, ésta intentará pasar al contenedor el control del evento de teclado.

a) Si recibe eventos de teclas de desplazamiento:

   i. Si el contenedor puede manejar el desplazamiento, la vista lo propagará.

   ii. Si el contenedor no puede manejar el evento, la vista intentará activar el siguiente componente visual

b) Si recibe otros eventos de tecla:

   i. La vista propagará el evento al contenedor.

### 5.3. Vista tiene un control de usuario como componente visual activo

Si en el paso 5, la vista determina que se encuentra activo un control de usuario, ésta intentará pasar al control el manejo del evento de teclado.

a) Si recibe eventos de teclas de desplazamiento:

   i. Si el control puede manejar el desplazamiento, la vista lo propagará.

   ii. Si el control no puede manejar el evento, la vista intentará activar el siguiente componente visual

b) Si recibe el evento de ejecución:

   i. La vista intentará propagar ese evento al control de usuario, que deberá manejarlo y ejecutar una acción.

c) Si recibe otro evento de teclado:

   i. La vista intentará propagar ese evento al control de usuario.

**6. Subflujos**

N/A

**7. Escenarios Clave**

**7.1. La vista no está compuesta por controles propios, contenedores u otros controles.**

1. La vista descartará el evento recibido.

**8. Post condiciones**

**8.1. Manejo exitoso del evento de teclado.**

El evento de teclado recibido, será manejado por alguna vista, contenedor o control de usuario. Eventualmente, el evento provocará que la vista sea repintada en la pantalla y se apreciarán cambios en los controles de usuario.

**9. Requerimientos especiales**

N/A

### 2.1.4. ACORDAR UN ENFOQUE TÉCNICO

El objetivo de esta actividad es definir una especificación técnica para el sistema, enmarcada dentro de las restricciones y condiciones establecidas previamente para el proyecto, que soporte adecuadamente sus requerimientos.

Durante esta actividad se crea un esbozo inicial del acercamiento técnico del sistema, se confirma que las decisiones técnicas son capturadas y difundidas de manera adecuada, y se asegura que hay suficiente información disponible para entender el enfoque seleccionado.

En esta actividad no se busca producir una especificación técnica demasiado detallada y exhaustiva del sistema, sino una que permita establecer un acercamiento técnico a alto nivel. Ésta especificación se refleja en un artefacto denominado Cuaderno de Arquitectura.

#### 2.1.4.1. DESARROLLO DE LA ACTIVIDAD

Para comenzar, se toma como insumos los artefactos generados principalmente durante la actividad denominada *Identificar y Refinar los Requerimientos*, es decir, la especificación de requerimientos globales, el documento de visión y el glosario.

El primer paso consiste en definir las metas de arquitectura para el proyecto. Estas metas permiten establecer la forma en que una solución debe responder a los cambios en el tiempo, así como las decisiones que podrían afectar el diseño de la solución, principalmente en función de los requerimientos del software.

Aplicación Práctica: UIL 4 MIDP

Para la solución materia de este análisis, se establecen las siguientes metas de arquitectura:

| No. | Metas de Arquitectura |
|---|---|
| 1 | Construir una biblioteca de clases que permita desarrollar MIDlets que tengan una vigencia de al menos un año calendario. |
| 2 | Lograr una biblioteca de clases óptima, desde el punto de vista del rendimiento y el consumo de recursos del sistema. |
| 3 | Minimizar las dependencias de la biblioteca de clases con el hardware de los dispositivos móviles. |
| 4 | Diseñar la biblioteca de clases de tal manera que permita a los desarrolladores de MIDlets empezar a trabajar con ella rápidamente y sin mayor dificultad. |

**Tabla 2.8. Metas de arquitectura para la solución.**
Elaborado por: Carlos Andrés Oquendo

El segundo paso es identificar los requerimientos funcionales que son más significativos desde el punto de vista de la arquitectura de la solución, es decir, aquellos que impactan directamente en la estructura básica de la solución y que deben ser resueltos primero.

La fuente de esta información es el artefacto denominado *Especificación de Requerimientos Globales del Sistema*, del cual se extraen específicamente los requisitos funcionales más relevantes, y se ordenan según su importancia y prioridad.

Aplicación Práctica: UIL 4 MIDP

De los requerimientos funcionales definidos para la solución, se han seleccionado los siguientes como más significativos:

| No. | Requerimiento |
|---|---|
| 1 | Proveer un conjunto básico de componentes visuales. |
| 2 | Incluir mecanismos que permitan organizar de manera automática los componentes visuales en la pantalla. |
| 3 | Incorporar mecanismos que faciliten la navegación y el desplazamiento entre las diferentes vistas de una aplicación. |

**Tabla 2.9. Requerimientos Arquitectónicos Significativos**
Elaborado por: Carlos Andrés Oquendo

A continuación se debe establecer y examinar las restricciones de arquitectura que aplican para el proyecto. En forma simultánea, se debe tomar algunas decisiones que van a definir su desarrollo.

Aplicación Práctica: UIL 4 MIDP

Para el proyecto, se han encontrado las siguientes restricciones de arquitectura:

| No. | Restricciones de Arquitectura |
|---|---|
| 1 | La solución tiene que usar el mínimo posible de recursos del sistema para dibujar las interfaces gráficas de un MIDlet en pantalla. |
| 2 | Es necesario que los componentes visuales produzcan una interfaz gráfica con apariencia actualizada y moderna. |
| 3 | No se deberá utilizar APIs específicas de los fabricantes de dispositivos móviles para implementar la solución. |
| 4 | La solución deberá implementarse para el uso en dispositivos con pantalla a color. |

**Tabla 2.10. Restricciones y Decisiones de Arquitectura**
Elaborado por: Carlos Andrés Oquendo

Las restricciones y decisiones señaladas, deben sustentarse en argumentos válidos para el proyecto. Éstos se verán reflejados en el artefacto correspondiente, que se genera al concluir con la actividad.

Posteriormente se procede a identificar las abstracciones claves para el sistema, en función de los requerimientos establecidos. Si no se ha identificado las abstracciones, es posible que se presenten complicaciones al momento de tratar de explicar la solución.

Open UP recomienda que en esta etapa del proyecto se realice una primera identificación de abstracciones y se incluya una descripción corta. Las abstracciones pueden evolucionar y cambiar a medida que el proyecto avanza y refinarse como elementos de diseño.

Aplicación Práctica: UIL 4 MIDP

En la Tabla 2.11 se listan las abstracciones clave identificadas para la solución. Su descripción se encontrará en el artefacto generado al final de esta actividad.

| No. | Nombre de la Abstracción Clave |
|---|---|
| 1 | Componente Visual |
| 2 | Vista |
| 3 | Ventana |
| 4 | Contenedor |
| 5 | Navegación |
| 6 | Pantalla |
| 7 | Apariencia |

**Tabla 2.11. Abstracciones Clave de la Solución**
Elaborado por: Carlos Andrés Oquendo

El siguiente paso consiste en validar las posibilidades de reutilización de algún elemento existente, como componentes, código fuente, patrones de diseño, etc., en función de la naturaleza de la solución.

Aplicación Práctica: UIL 4 MIDP

Para la construcción de la biblioteca de clases, no se va a reutilizar ningún componente complejo previamente elaborado por terceros fabricantes, sino que la solución va a construirse en base al API de bajo nivel presente en el perfil MIDP y en la configuración CLDC.

Para la implementación de algunas funcionalidades específicas, como el manejo de colores y gradientes, se podrá reutilizar código genérico disponible en Internet, modificándolo y adaptándolo a las necesidades específicas del proyecto, para asegurar su correcto funcionamiento.

Para implementar algunos mecanismos y funcionalidades de la solución, se considerará la posibilidad de utilizar algunos patrones de diseño y programación comunes, como por ejemplo, fábricas *(factory)*, métodos de fábrica, constructores *(builder), singleton*, etc.

También es necesario definir si la solución debe particionarse o dividirse en capas. Este proceso depende, nuevamente, de la naturaleza de la solución.

Aplicación Práctica: UIL 4 MIDP

En general, se puede decir que las bibliotecas de clases conforman una capa de servicios genéricos que una determinada aplicación está en capacidad de utilizar y aprovechar.

Considerando que el proyecto comprende la construcción de una biblioteca de clases, se ha resuelto no establecer una división en capas. Lo que si se tendrá en cuenta es la separación interna de funcionalidades, para lo cual se utilizará una estructura de paquetes adecuada, según las especificaciones del lenguaje de programación Java.

Para continuar con el proceso, se debe especificar los mecanismos de arquitectura aplicables para la solución. Los mecanismos de arquitectura son soluciones comunes, para problemas comunes, que pueden ayudar a reducir la complejidad de la solución.

Es recomendable elaborar una descripción de cada mecanismo de arquitectura identificado. Posteriormente, se puede detallar cada mecanismo hasta llegar a realizar una implementación.

Aplicación Práctica: UIL 4 MIDP

Para el proyecto, se han identificado los siguientes mecanismos de arquitectura:

| No. | Mecanismo de Arquitectura |
|---|---|
| 1 | Pintado de Gradientes |
| 2 | Redimensionamiento |
| 3 | Selección de componentes |
| 4 | Cambio de vistas |
| 5 | Organización de componentes |

**Tabla 2.12. Mecanismos de Arquitectura de la Solución**
Elaborado por: Carlos Andrés Oquendo

La descripción de estos mecanismos se visualizará en el artefacto producto de esta actividad.

A continuación, se debe identificar las interfaces a los sistemas externos, de hardware o de software, con los cuales la solución tendrá que interactuar.

Aplicación Práctica: UIL 4 MIDP

La solución se relaciona principalmente con dos sistemas de software:

- Por un lado, se encuentra el entorno en tiempo de ejecución Java Micro Edition disponible en cada uno de los dispositivos móviles, y específicamente el perfil MIDP, que brinda los servicios básicos para el funcionamiento de la solución.
- Por otro lado, se encuentra la aplicación que se construye utilizando la solución y que consume los servicios provistos por ésta.

Finalmente, se debe plasmar las decisiones de arquitectura tomadas durante esta actividad en el artefacto de resultados correspondiente, que sirve como referencia futura para los interesados.

Este proceso debe ser realizado de tal manera que lo que se incluya en el artefacto se mantenga consistente con los requisitos de la solución.

#### 2.1.4.2. ARTEFACTOS

Cuaderno de Arquitectura

Este artefacto permite describir las razones, suposiciones, explicaciones e implicaciones de las decisiones que se tomaron al momento de diseñar la arquitectura de la solución.

Sin este artefacto la coordinación de los esfuerzos de diseño individuales se dificulta, así como el establecimiento y la comunicación de una visión arquitectónica general para el proyecto.

En el caso de esta solución, el cuaderno de arquitectura que se ha desarrollado es el siguiente:

# User Interface Library for MIDP - UIL 4 MIDP
# Cuaderno de Arquitectura

## 1. Propósito

Este documento describe la filosofía, las decisiones, restricciones, justificaciones, elementos significativos y demás aspectos generales del sistema que delinean el diseño y la implementación.

## 2. Metas arquitectónicas

- Lograr una biblioteca de clases óptima, desde el punto de vista del rendimiento y el consumo de recursos del sistema.
- Minimizar la dependencia de la solución del hardware subyacente.
- Diseñar los componentes de la solución de una manera lógica y coherente, que permita a los desarrolladores empezar a trabajar con ellos rápidamente y sin mayor dificultad.

## 3. Suposiciones y dependencias

- Se asume que los dispositivos objetivos de la solución se basan en el perfil MIDP 2.0 y en la configuración CLDC 1.1.
- Se debe conocer el lenguaje de programación Java, de preferencia, con un nivel medio de experiencia.
- Se asume que se tendrán disponibles los SDK de varios fabricantes para probar la solución genérica.

## 4. Requisitos arquitectónicos significativos

De los requerimientos enumerados en la sección 2 del documento denominado *UIL 4 MIDP: Especificación de Requerimientos Globales del Sistema*, se han seleccionado los siguientes como requisitos arquitectónicos significativos.

- Proveer un conjunto básico de componentes visuales.
- Incluir mecanismos que permitan organizar de manera automática los componentes visuales en la pantalla.
- Incorporar mecanismos que faciliten la navegación y el desplazamiento entre las diferentes vistas de una aplicación.

## 5. Decisiones, restricciones y justificaciones

- La solución tiene que usar el mínimo posible de recursos del sistema para dibujar las interfaces gráficas de un MIDlet en pantalla.
    - Los dispositivos móviles donde la solución deberá funcionar, como parte fundamental de un MIDlet, tienen recursos del sistema reducidos y usualmente limitados por el fabricante.

- Es necesario que los componentes visuales produzcan una interfaz gráfica con apariencia actualizada y moderna.
    - El éxito de la solución radica en permitir la creación de interfaces gráficas para MIDlets que sean actualizadas y modernas. Para esto, los componentes visuales de la solución deberán tener algunas características como una apariencia predeterminada rediseñada, el ajuste automático a los tamaños de pantalla, la organización automática en las diferentes vistas y facilidades para ordenar la navegación entre las vistas de una aplicación.

- No se deberá utilizar APIs específicas de los fabricantes de dispositivos móviles para implementar la solución.
    - La solución debe construirse en base a las funcionalidades básicas disponibles en la plataforma Java Micro Edition, específicamente, en el perfil MIDP y en la configuración CLDC, para soportar un amplio rango de dispositivos.
    - De la misma manera, no se soportará de manera nativa otras plataformas basadas en Java Micro Edition a las que los fabricantes de dispositivos hayan hecho adaptaciones específicas, como por ejemplo, la plataforma BlackBerry, desarrollada por Research In Motion.

- La solución deberá implementarse para el uso en dispositivos con pantalla a color.
    - Los dispositivos que incluyen pantalla monocromática usualmente son plataformas cerradas de gama baja, que no permiten la instalación de aplicaciones creadas por terceros fabricantes.
    - El uso de los colores permite crear interfaces gráficas más llamativas y atractivas para los usuarios. La aplicación de esta solución en un dispositivo con pantalla monocromática sería irrelevante ya que no se puede aprovechar la riqueza visual que la solución debe proporcionar para la creación de interfaces gráficas de usuario para MIDlets.

## 6. Mecanismos de arquitectura

### Pintado de Gradientes

- Permite rellenar una región definida con las variaciones de tono entre dos o más colores específicos.

### Redimensionamiento

- Permite determinar y establecer valores para el tamaño que posee un componente respecto a su contenedor, en función del tamaño de la pantalla en el que se dibujen las interfaces gráficas de una aplicación.

### Selección de componentes

- Permite establecer el estado de activación de un único componente perteneciente a una vista o contenedor. Cuando un componente se encuentra seleccionado, es capaz de interactuar y responder a los eventos generados por el usuario.

### Cambio de vistas

- Se encarga de gestionar el modo en el que una vista, que se encuentra en estado visible, es reemplazada por otra vista diferente, ante la ejecución de un evento generado por el usuario en la primera vista o en vistas externas.

### Organización de componentes

- Define la forma en la que los componentes pertenecientes a un contenedor son organizados cuando se presentan en la pantalla del dispositivo, como parte de la interfaz gráfica de una aplicación.

## 7. Abstracciones clave

- *Componente Visual:* Son el conjunto de elementos que permiten componer las diferentes interfaces gráficas de usuario de un MIDlet. Los componentes visuales, que también se pueden denominar *Controles*, responden usualmente a eventos generados por los usuarios.
- *Vista:* Conjunto de controles que permiten mostrar información relevante al usuario para que éste realice alguna acción.
- *Ventana:* Medio en el cual se despliegan las diferentes vistas de un MIDlet.
- *Contenedor:* Componente que puede albergar a otros controles. Se debe usar para organizar los controles de una vista.
- *Navegación:* Desplazamiento entre las diversas vistas de las que se compone la interfaz gráfica de un MIDlet. La navegación se produce cuando el usuario, al disparar un evento en una vista, hace que la aplicación cambie de vista activa.
- *Pantalla:* Periférico parte de los dispositivos móviles por el cual se muestra información a los usuarios.
- *Apariencia:* Conjunto de valores que definen la forma en que se pintan los controles en las vistas pertenecientes a la interfaz gráfica de un MIDlet.

## 8. Capas o marco de trabajo arquitectónico

Esta solución no se va a diseñar ni a implementar usando como base un modelo de capas.

La solución se generará como una biblioteca de clases, que usualmente se referencia completa en las aplicaciones que la consumen. Cuando se construya una aplicación en base a la solución, ésta puede aparecer como una capa de servicios genéricos para la aplicación.

## 9. Vistas arquitectónicas

En el siguiente diagrama se aprecia la relación de la solución UIL 4 MIDP, con los MIDlets, en un ambiente basado en Java Micro Edition, MIDP y CLDC:

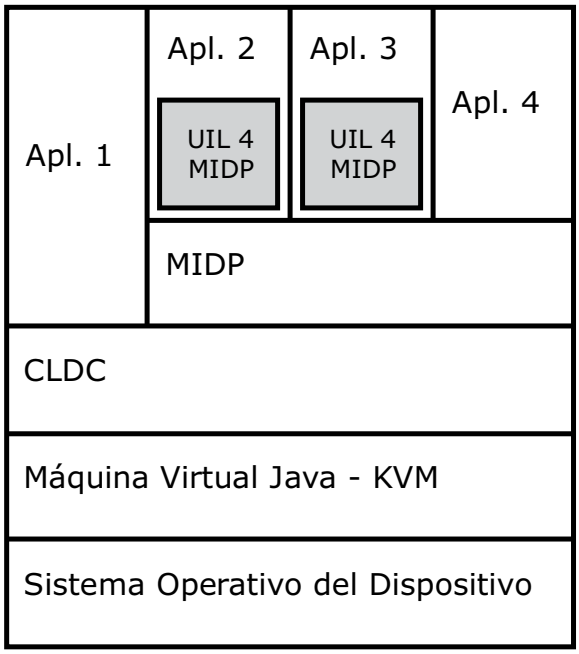



**Figura 9-1. Relación de la solución en el ambiente Java ME**
Elaborado por: Carlos Andrés Oquendo

Podemos apreciar que para cada MIDlet, la solución UIL 4 MIDP se puede incluir como componente interno de la aplicación.

La solución, internamente, se incorporará los siguientes elementos:

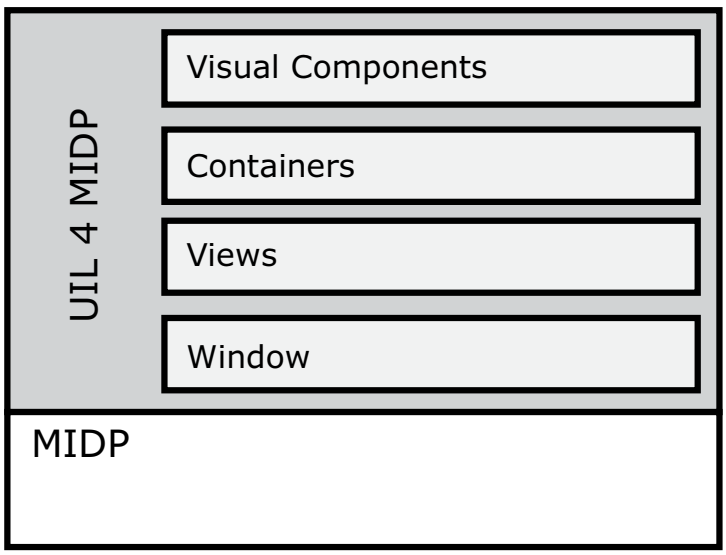



**Figura 9-2. Elementos internos de la solución**
Elaborado por: Carlos Andrés Oquendo

En el grupo de *Visual Components* se encuentran todos los componentes o controles de usuario con los cuales es posible la interacción directa del usuario, solicitando y mostrando información, y manejando eventos.

En el grupo de *Containers* se encuentran los componentes que permiten agrupar y organizar los controles según la necesidad de los desarrolladores. Dentro de este grupo se encuentran además componentes que permiten gestionar la disposición de los controles en las diferentes vistas.

En el grupo de *Views* se encuentran algunos componentes que se pueden mostrar directamente en la ventana de la aplicación y que pueden incluir controles del grupo de Visual Components o elementos del grupo de Containers.

*Window* representa al componente capaz de mostrar vistas en la pantalla del dispositivo.

## 2.2. FASE DE ELABORACIÓN

En la fase de elaboración, el esfuerzo que se debe realizar se enfoca en lograr los siguientes objetivos:

- Entender con más detalle los requisitos
- Diseñar, implementar, validar y fundamentar una arquitectura
- Mitigar riesgos esenciales y producir estimados precisos de costos y tiempos.

Las actividades que se enumeran y describen en los siguientes puntos permiten cumplir con los objetivos expuestos.

### 2.2.1. PLANEAR Y GESTIONAR LAS ITERACIONES

Esta actividad se define de la misma manera que se hizo para el punto 2.1.2 Planear y Gestionar las Iteraciones, relacionado con la fase de inicio del proyecto.

La diferencia radica únicamente en que para esta fase del proyecto ya se dispone de una base de información previamente desarrollada y asentada en los artefactos Lista de Riesgos y Lista de Elementos de Trabajo, que deben ser refinados y actualizados a medida que transcurre el proyecto.

#### 2.2.1.1. DESARROLLO DE LA ACTIVIDAD

Para ejecutar esta actividad se debe revisar los artefactos mencionados y validar si no existen riesgos o elementos de trabajo que no hayan sido consideradas.

Aplicación Práctica: UIL 4 MIDP

De la revisión realizada, se han detectado algunos riesgos no contemplados en la fase anterior, que se enumeran en la Tabla 2.13.

| No. | Riesgos |
|---|---|
| 1 | Distracción del recurso humano |
| 2 | Desconocimiento de mecanismos para la implementación |

**Tabla 2.13. Riesgos adicionales identificados para el proyecto**
Elaborado por: Carlos Andrés Oquendo

Los nuevos riesgos, sin embargo, no alteran mucho la lista de elementos de trabajo. La nueva actividad que se añade se puede realizar en paralelo a las actividades previamente identificadas. Ésta actividad se lista en la Tabla 2.14.

| No. | Actividades |
|---|---|
| 1 | Investigar el funcionamiento de mecanismos y elementos desconocidos |

**Tabla 2.14. Actividad adicional detectada para el proyecto.**
Elaborado por: Carlos Andrés Oquendo

Hay que destacar que ambos artefactos actualizados, al igual que el diagrama de Gantt, se incluirán como anexos al final del documento.

Como ya se dispone de información más afinada y precisa de las tareas requeridas y sus duraciones, se puede establecer un presupuesto referencial para el proyecto.

Aplicación Práctica: UIL 4 MIDP

Para determinar el potencial costo del proyecto UIL4MIDP se analizará los diferentes elementos que intervienen en su desarrollo.

*1. Recursos Humanos*

Los costos de recursos humanos se calculan en función de los diferentes roles que se deben asumir a lo largo de la vida del proyecto. Para cada uno de los roles se ha establecido el tiempo con el que se debe contar y el valor que en un periodo de tiempo determinado se espera consignar.

| Cargo / Rol | Horas en Proyecto | Costo por Hora | Total |
|---|---|---|---|
| **Administrador del Proyecto** | 240 | $ 4,50 | $ 1.080,00 |
| **Analista del Sistema** | 120 | $ 5,00 | $ 600,00 |
| **Desarrollador y Tester** | 320 | $ 3,50 | $ 1.120,00 |
| **Diseñador del Sistema** | 240 | $ 6,00 | $ 1.440,00 |
| **Total Recursos Humanos** | | | **$ 4.240,00** |

**Tabla 2.15. Costo de Recursos Humanos del Proyecto**
Elaborado por: Carlos Andrés Oquendo

*2. Hardware y Equipos*

Se asume que se debe alquilar el hardware necesario para realizar el desarrollo. De esta manera, se libera la responsabilidad de proyectar y distribuir el costo de los equipos entre varios proyectos futuros.

En la Tabla 2.16 se determinan los equipos requeridos y su cantidad, el costo del alquiler en un periodo de tiempo determinado y el tiempo de uso aproximado.

| Equipo | Cant. | Concepto | Unidad | Valor Unitario | Tiempo | Total |
|---|---|---|---|---|---|---|
| **Computador Portátil para Desarrollo** | 1 | Alquiler | Mes | $ 100,00 | 4 | $ 400,00 |
| **Teléfonos Celulares para Pruebas** | 9 | Alquiler | Hora | $ 10,00 | 2 | $ 180,00 |
| **Total Hardware y Equipos** | | | | | | **$ 580,00** |

**Tabla 2.16. Costo de Hardware y Equipos del Proyecto**
Elaborado por: Carlos Andrés Oquendo

*3. Software*

El costo del software se calcula en función del uso que tendrá en el proyecto respecto a su vida útil. Para cada producto se indica el número de licencias.

Además, algunos de los productos que se utilizarán durante el desarrollo de la solución no tienen costo, debido a que los fabricantes los liberan como herramientas gratuitas, o porque son productos de software libre.

Los valores incluidos en la Tabla 2.17 fueron obtenidos en el sitio web PriceGrabber que muestra, de manera condensada, la información de un producto disponible en múltiples sitios de comercio electrónico a nivel mundial.

| Paquete | Cantidad de Licencias | Valor Unitario | Valor Total | Utilización (Porcentaje) | Total Para Proyecto |
|---|---|---|---|---|---|
| **Sybase PowerDesigner** | 1 | $ 2.800,00 | $ 2.800,00 | 10% | $ 280,00 |
| **Adobe Photoshop CS5** | 1 | $ 250,00 | $ 250,00 | 20% | $ 50,00 |
| **NetBeans IDE 6.9.1** | 1 | $ - | $ - | 100% | $ - |
| **Microsoft Office 2010** | 1 | $ 220,00 | $ 220,00 | 25% | $ 55,00 |
| **Microsoft Project 2010** | 1 | $ 367,00 | $ 367,00 | 15% | $ 55,05 |
| **Java ME SDK** | 1 | $ - | $ - | 100% | $ - |
| **Nokia S40 SDK** | 1 | $ - | $ - | 100% | $ - |
| **Total Software** | | | | | **$ 440,05** |

**Tabla 2.17. Costo de Software para el Proyecto**
Elaborado por: Carlos Andrés Oquendo
Fuente: http://www.pricegrabber.com

*4. Otros Costos*

Además de los recursos humanos, el hardware y el software, se requieren de otros productos y servicios para el correcto desarrollo del proyecto.

Para el cálculo de los valores totales, se debe considerar el tiempo de potencial uso del producto o servicio

| Descripción | Tiempo (Meses) | Valor Unitario | Total |
|---|---|---|---|
| **Internet Banda Ancha 1 Mbps** | 4 | $ 24,90 | $ 99,60 |
| **Energía Eléctrica** | 4 | $ 40,00 | $ 160,00 |
| **Servicios de Telecomunicaciones** | 4 | $ 15,00 | $ 60,00 |
| **Otros** | 4 | $ 20,00 | $ 80,00 |
| **Movilización** | 4 | $ 15,00 | $ 60,00 |
| **Total Otros Costos y Gastos** | | | **$ 459,60** |

**Tabla 2.18. Costo de Otros Productos y Servicios para el Proyecto**
Elaborado por: Carlos Andrés Oquendo

*5. Resumen*

Una vez determinados los valores parciales para cada una de las categorías de cálculo, es necesario condensarlos para obtener un estimado del valor total del proyecto.

Además de los valores obtenidos, se incluyó un rubro correspondiente al cálculo de la utilidad para el proyecto.

**Resumen de Presupuesto Referencial del Proyecto**
***User Interface Library for MIDP - UIL4MIDP***

| Descripción | | Total |
|---|---|---|
| Recursos Humanos | | $ 4.240,00 |
| Hardware y Equipos | | $ 580,00 |
| Software | | $ 440,05 |
| Otros Costos y Gastos | | $ 459,60 |
| **Total Costos** | | **$ 5.719,65** |
| Utilidad | 15% | $ 857,95 |
| **Precio Total del Proyecto** | | **$ 6.577,60** |

**Tabla 2.19. Resumen del Presupuesto del Proyecto**
Elaborado por: Carlos Andrés Oquendo

En los entregables correspondientes se incluye el libro de trabajo donde se realizó los diferentes cálculos del costo y precio del proyecto.

### 2.2.2. IDENTIFICAR Y REFINAR LOS REQUERIMIENTOS

En esta fase, el enfoque de esta actividad, que se realiza por primera vez en la fase de inicio, cambia y se orienta a definir la solución. Esta consiste en encontrar los requisitos que son más valiosos desde el punto de vista de los interesados, que son particularmente desafiantes o riesgosos, o que son significativos para la arquitectura.

Los requerimientos deben ser priorizados y detallados de manera que sean completamente entendibles para los desarrolladores y para los interesados.

Una vez que se logra un adecuado nivel de detalle para los requisitos, se puede comenzar a implementar el software.

#### 2.2.2.1. DESARROLLO DE LA ACTIVIDAD

Esta actividad se desarrolla tomando como fundamento al artefacto denominado Requerimientos Globales del Sistema, desarrollado durante la fase de inicio del proyecto de desarrollo.

La primera tarea que se realiza es identificar requerimientos y delinearlos. Es posible que durante la fase de inicio, se hayan omitido, voluntaria o involuntariamente, algunos requerimientos importantes para el sistema, o que algún interesado no haya expresado sus expectativas y necesidades completamente.

Por estos motivos, es conveniente validar el alcance de la solución con los interesados y determinar si éste se encuentra completo o no. En el caso de no estar completo, es necesario establecer los requerimientos faltantes y proceder a actualizar el artefacto correspondiente.

| Aplicación Práctica: UIL 4 MIDP |
| --- |
| Para el caso de la solución, la revisión del alcance establecido en la fase de inicio no ha arrojado requerimientos faltantes o incompletos, por lo que no es necesario actualizar por el momento el artefacto asociado. |

Es también necesario proceder con el detalle de los requerimientos existentes en el artefacto de requerimientos globales.

Aplicación Práctica: UIL 4 MIDP

Los pormenores de cada uno de los requerimientos funcionales se aprecian en la Tabla 2.20, que aparece a continuación.

| No. | Requerimiento Funcional |
|---|---|
| 1 | Proveer un conjunto básico de componentes visuales |
| | Se deberán proveer al menos los siguientes componentes:<br><br>▪ Label (Etiqueta) – Muestra texto que los usuarios no pueden modificar.<br>▪ TextBox (Cuadro de Texto) – Permite a los usuarios ingresar o modificar el texto que contiene el control.<br>▪ ImageBox (Cuadro de Imagen) – Permite a los usuarios visualizar una imagen.<br>▪ Switch (Interruptor) – Permite establecer el valor de activo o inactivo para una opción determinada.<br>▪ OptionPicker (Selector de Opciones) – Permite a los usuarios escoger una de varias opciones disponibles.<br>▪ Button (Botón) – Ejecuta una acción determinada.<br>▪ StackedContainer (Contenedor Apilado) – Permite agrupar varios componentes con un orden específico en forma vertical. Las medidas y dimensiones de los componentes agrupados se establecen de manera automática.<br>▪ HorizontalSplittedContainer (Contenedor Dividido Horizontal) – Permite agrupar dos componentes y ubicarlos uno junto al otro en forma horizontal. Dependiendo de la configuración, el contenedor puede asignar anchos absolutos o relativos a los controles contenidos. |
| 2 | Incluir mecanismos que permitan organizar de manera automática los componentes visuales en la pantalla |
| | Se incorporarán los siguientes mecanismos de organización<br><br>▪ Organización Horizontal – Los controles se ordenarán uno tras de otro en fila. El ancho del contenedor padre se repartirá entre todos los controles agrupados.<br>▪ Organización Vertical – Los controles se ordenarán de arriba hacia abajo como en una lista. Esta será la organización predeterminada. |
| 3 | Incorporar mecanismos que faciliten la navegación y el desplazamiento entre las diferentes vistas de una aplicación |
| | Se incorporarán los siguientes mecanismos de navegación.<br><br>▪ Navegación jerárquica – Basada en listas de opciones. Para cada opción de una lista puede haber definida otra lista de opciones o una vista que se asume como punto final de la navegación. Para volver a la raíz se debe pasar por todos los niveles definidos.<br>▪ Navegación por pestañas – Basada en vistas independientes. Se incluye un medio para cambiar indistintamente entre las diferentes vistas.<br>▪ Navegación libre – Basada en vistas independientes. Cada vista debe incluir botones que permitan cambiar a otra vista. |
| 4 | Introducir mecanismos que permitan la personalización de los componentes visuales |
| | Los componentes deberán soportar el ajuste de al menos las siguientes características:<br><br>▪ Dimensiones (ancho, alto) – Manejo automático<br>▪ Márgenes de los contenedores |

| | |
|---|---|
| | ▪ Relleno (margen interno) de los componentes<br>▪ Color principal (usualmente el color del texto)<br>▪ Color de fondo (color sólido o degradado)<br><br>El texto de los controles podrá personalizarse al menos de las siguientes maneras:<br><br>▪ Negrilla<br>▪ Subrayado<br>▪ Cursiva |
| 5 | Establecer mecanismos que permitan a los controles ajustarse automáticamente al tamaño de la pantalla del dispositivo móvil |
| | Se deberá incorporar las siguientes funcionalidades:<br><br>▪ Detección de las dimensiones de la pantalla.<br>▪ Ajuste del ancho de los controles en función del ancho de la pantalla<br>▪ Visualización de controles en función del alto de la pantalla.<br>▪ Mecanismo de desplazamiento de los controles (scrolling) |
| 6 | Los componentes de la solución deben tener la capacidad de reconocer eventos para la aplicación |
| | Los componentes deberán soportar el reconocimiento de los siguientes eventos:<br>▪ Enfoque – Se produce cuando un control obtiene el foco de otro control.<br>▪ Selección – Se produce cuando el usuario utiliza el control, es decir, desencadena las acciones que el control tiene especificadas.<br>▪ Desenfoque – Se produce cuando un control pasa el foco a otro control. |

**Tabla 2.20. Detalle de Requerimientos Funcionales**
Elaborado por: Carlos Andrés Oquendo

Si en el detalle de requerimientos aparecen nuevos términos y definiciones, deben incluirse en el glosario.

Finalmente, se vuelve necesario revisar el esquema de licenciamiento implementado para asegurarse de su compatibilidad con la solución.

Aplicación Práctica: UIL 4 MIDP

En la Fase de Inicio se estableció que UIL4MIDP se licenciaría bajo un esquema dual y se definió que, de las dos necesarias, la primera sería la licencia de software libre Apache 2.0.

Para definir la licencia privativa, se hizo una revisión rápida de los contratos de licencia de usuario final de algunos productos licenciados bajo este esquema dual, y se procedió a redactar la siguiente propuesta de contrato de licencia:

**CONTRATO DE LICENCIA DE USUARIO FINAL PARA "USER INTERFACE LIBRARY FOR MIDP" (UIL 4 MIDP).**

**IMPORTANTE**. POR FAVOR LEA CON DETENIMIENTO LOS TÉRMINOS Y CONDICIONES DE ESTE ACUERDO DE LICENCIA ANTES DE UTILIZAR ESTE PROGRAMA:

El Contrato de Licencia de Usuario Final (CLUF) de la ESCUELA POLITÉCNICA NACIONAL es un acuerdo legal entre USTED (una persona natural o jurídica) y la ESCUELA POLITÉCNICA NACIONAL para los productos de software de la ESCUELA POLITÉCNICA NACIONAL identificados anteriormente, los cuales pueden incluir componentes de software asociados, material impreso, multimedia y documentación digital o electrónica, que en adelante se denominarán PRODUCTO DE SOFTWARE. Al instalar, copiar o utilizar el PRODUCTO DE SOFTWARE, USTED acepta los términos de este CLUF. Este contrato de licencia representa todo el acuerdo relativo al programa entre USTED y la ESCUELA POLITÉCNICA NACIONAL, que en adelante se denominará el LICENCIATARIO, y reemplaza cualquier propuesta, representación o entendimiento previo entre las partes. Si USTED no acepta los términos de este CLUF, no instale o use el PRODUCTO DE SOFTWARE.

El PRODUCTO DE SOFTWARE se encuentra protegido por las leyes de propiedad intelectual y tratados de derechos de autor de la República del Ecuador e internacionales, así como por cualquier otras leyes y tratados de propiedad intelectual aplicables. El PRODUCTO DE SOFTWARE se licencia, no se vende.

1. CONCESIÓN DE LICENCIA
   El PRODUCTO DE SOFTWARE se licencia de la siguiente manera:
   1.1. INSTALACIÓN Y USO
   La ESCUELA POLITÉCNICA NACIONAL le concede a USTED el derecho de instalar o usar copias del PRODUCTO DE SOFTWARE en los equipos y dispositivos que ejecuten una plataforma para la cual el PRODUCTO DE SOFTWARE fue diseñado (Java Micro Edition, MIDP 2.0, CLDC 1.1). Podrá usar el PRODUCTO DE SOFTWARE para construir aplicaciones basadas en él. Cualquier aplicación construida en base al PRODUCTO DE SOFTWARE podrá redistribuir, en forma embebida, una copia sin modificaciones de él para uso exclusivo de la aplicación.
   1.2. COPIAS DE RESPALDO
   USTED puede también hacer copias del PRODUCTO DE SOFTWARE que considere necesarias para propósitos de respaldo y archivo.

2. DESCRIPCIÓN DE OTROS DERECHOS Y LIMITACIONES
   2.1. MANTENIMIENTO DE NOTAS DE DERECHOS DE AUTOR
   USTED no puede remover o alterar cualquier nota de derechos de autor en cualquiera de las copias del PRODUCTO DE SOFTWARE que tenga a su disposición.
   2.2. DISTRIBUCIÓN
   USTED no puede distribuir las copias registradas a su nombre del PRODUCTO DE SOFTWARE a terceras partes.
   Si la ESCUELA POLITÉCNICA NACIONAL pone versiones de evaluación del PRODUCTO DE SOFTWARE disponibles para descarga en su(s) sitio(s) web, y USTED tiene acceso a ellas, podrá redistribuir libremente estas versiones siempre y cuando se mantengan intactas.
   2.3. PROHIBICIÓN DE INGENIERÍA REVERSA, DESCOMPILACIÓN Y DESENSAMBLADO
   USTED no puede aplicar técnicas de ingeniería reversa, ni procedimientos de

descompilación y de desensamblado del PRODUCTO DE SOFTWARE, excepto y solamente cuando esa actividad sea permitida expresamente por la ley aplicable, a pesar de esta limitación.

2.4. RENTA

USTED no puede rentar, arrendar o prestar el PRODUCTO DE SOFTWARE que le haya sido licenciado.

2.5. SERVICIOS DE SOPORTE

La ESCUELA POLITÉCNICA NACIONAL puede proveerle Servicios de Soporte relacionados al PRODUCTO DE SOFTWARE. Cualquier código suplementario provisto como parte de los Servicios de Soporte debe considerarse parte del PRODUCTO DE SOFTWARE y está sujeto a las condiciones de este CLUF.

2.6. CUMPLIMIENTO DE LAS LEYES APLICABLES

USTED debe cumplir con todas las leyes aplicables relacionadas con el uso del PRODUCTO DE SOFTWARE.

3. TERMINACIÓN

Sin prejuicio de cualquier otro derecho, la ESCUELA POLITÉCNICA NACIONAL puede terminar este CLUF si usted no cumple con los términos y condiciones de este CLUF. En ese evento, usted DEBE destruir todas las copias del PRODUCTO DE SOFTWARE en su poder.

4. DERECHOS DE AUTOR

Todo título, incluyendo pero no limitado a los Derechos de Autor, del PRODUCTO DE SOFTWARE y cualquier copia de él, son propiedad de la ESCUELA POLITÉCNICA NACIONAL. Todos los títulos y derechos de propiedad intelectual de las aplicaciones construidas en base al PRODUCTO DE SOFTWARE pertenecen a sus respectivos dueños. Todos los títulos y derechos de propiedad intelectual relacionados al contenido que se pueda acceder a través de las aplicaciones desarrolladas en base al PRODUCTO DE SOFTWARE son propiedad de sus respectivos dueños y este CLUF no le concede derechos para usar ese contenido.

Tanto el PRODUCTO DE SOFTWARE, como las aplicaciones basadas en él y el contenido accedido a través de las aplicaciones desarrolladas en base a él, pueden estar protegidos por derechos de autor aplicables u otros tratados y leyes de propiedad intelectual.

Todos los derechos no concedidos expresamente en este CLUF se encuentran reservados por la ESCUELA POLITÉCNICA NACIONAL.

5. RESTRICCIÓN DE GARANTÍA

La ESCUELA POLITÉCNICA NACIONAL expresamente renuncia a proveer cualquier tipo de garantía respecto al PRODUCTO DE SOFTWARE. El PRODUCTO DE SOFTWARE se provee ‘tal cual’ sin ninguna garantía expresa o implícita de cualquier tipo, incluyendo pero no limitada, a cualquier garantía de comerciabilidad, no infracción o acondicionamiento de un propósito particular. La ESCUELA POLITÉCNICA NACIONAL no garantiza o asume responsabilidades acerca de la precisión o completitud de cualquier información, texto, gráficos, enlaces u otros elementos contenidos dentro del PRODUCTO DE SOFTWARE. La ESCUELA POLITÉCNICA NACIONAL no hace ninguna garantía respecto a cualquier afectación que pueda ser causada por la transmisión de software malintencionado (como virus de computadora, gusanos, bombas de tiempo, bombas lógicas, o cualquier programa similar). La ESCUELA POLITÉCNICA NACIONAL además renuncia expresamente a proveer cualquier tipo de garantía o representación a Usuarios Autorizados o a cualquier tercera parte.

6. LIMITACIÓN DE RESPONSABILIDAD

La ESCUELA POLITÉCNICA NACIONAL no podrá, en ningún caso, ser responsable por cualquier tipo de daño (incluyendo y sin limitación a pérdidas de ingresos, lucro cesante,

interrupción de negocios, o información perdida) que se produzca por el uso o la incapacidad de uso del PRODUCTO DE SOFTWARE por parte de los Usuarios Autorizados, incluso cuando la ESCUELA POLITÉCNICA NACIONAL haya sido notificado de la posibilidad de esos daños. En ningún caso la ESCUELA POLITÉCNICA NACIONAL será responsable por pérdidas de datos o por los daños indirectos, especiales, incidentales o casuales (incluyendo lucro cesante) u otros daños basados en contrato, agravio o por cualquier otra causa. La ESCUELA POLITÉCNICA NACIONAL no se responsabiliza con respecto al contenido del PRODUCTO DE SOFTWARE o de parte de él, incluyendo pero no limitado a errores u omisiones que en él se incluyen, difamación, violación de los derechos de publicidad, privacidad, derechos de marca, interrupción de negocios, lesiones personales, pérdida de la privacidad, derechos morales o de la divulgación de información confidencial.

### 2.2.3. DESARROLLAR LA ARQUITECTURA

Esta actividad permite refinar la arquitectura de alto nivel para producir software funcional y estable que mitigue los riesgos técnicos del alcance.

Lo que se busca es refinar el bosquejo inicial de la arquitectura y producir elementos concretos de diseño, asegurándose que las directrices de arquitectura se han capturado y comunicado adecuadamente, comprobando que el equipo tiene suficiente información para desarrollar el software y, garantizando que los requisitos son correctamente gestionados.

#### 2.2.3.1. DESARROLLO DE LA ACTIVIDAD

La actividad comienza tomando como insumos a la especificación técnica para el proyecto y al cuaderno de arquitectura, elaborado al final de la fase de inicio.

El primer paso consiste en refinar las metas de arquitectura y los requerimientos significativos desde el punto de vista de la arquitectura. Como parte de este proceso, se debe verificar la existencia de requisitos significativos no contemplados, así como el cambio del enfoque de las metas de arquitectura previamente definidas.

Aplicación Práctica: UIL 4 MIDP

En el caso de la solución, no se han incorporado nuevos requerimientos significativos, y las metas de arquitectura se mantienen estables.

El siguiente paso es refinar los mecanismos de arquitectura previamente definidos para la solución. Este proceso puede permitir la inclusión de elementos tecnológicos y se puede representar como un patrón de diseño o como código fuente de ejemplo, dependiendo de las funcionalidades requeridas para la solución.

Aplicación Práctica: UIL 4 MIDP

Luego de revisar los mecanismos de arquitectura definidos en la Fase de Inicio, se ha decidido reestructurar el mecanismo denominado “Organización de componentes”, y renombrarlo “Inclusión y Organización de componentes”. De la misma forma, se ha cambiado el nombre del mecanismo antes denominado “Redimensionamiento” a “Dimensionamiento”.

La descripción y detalle de todos los mecanismos se incluye a continuación.

**Pintado de Gradientes:** Permite rellenar una región definida con las variaciones de tono entre dos o más colores específicos.

La variación de los tonos de los colores debe hacerse progresivamente para lograr que la mezcla de colores se produzca de manera uniforme. En general, el mecanismo debe soportar al menos dos direcciones de pintado: Horizontal y Vertical.

La implementación de este mecanismo se encuentra realizada en el código fuente disponible en el sitio web de Nokia Developer, en la dirección http://developer.nokia.com/Community/Wiki/Draw_Gradient_in_Java_ME

Una vez revisado el código fuente, y tomando en cuenta que la inclusión de componentes o código de terceros fomenta la reusabilidad de software, cuando es aplicable, se ha decidido utilizar esta implementación. El código fuente se lista a continuación:

```
public static void paintGradient(Graphics g, int startColor,
        int endColor, int left, int top,
        int width, int height, int orientation) {
    int max = orientation == VERTICAL ? height : width;

    for (int i = 0; i < max; i++) {
        int color = midColor(startColor, endColor,
                max * (max - 1 - i) / (max - 1), max);
```

```
            g.setColor(color);

            if (orientation == VERTICAL) {
                g.drawLine(left, top + i, left + width - 1, top + i);
            } else {
                g.drawLine(left + i, top, left + i, top + height - 1);
            }
        }
    }

    static int midColor(int startColor, int endColor, int prop, int max) {
        int red = (((startColor >> 16) & 0xff) * prop
                + ((endColor >> 16) & 0xff) * (max - prop)) / max;

        int green = (((startColor >> 8) & 0xff) * prop
                + ((endColor >> 8) & 0xff) * (max - prop)) / max;

        int blue = (((startColor >> 0) & 0xff) * prop
                + ((endColor >> 0) & 0xff) * (max - prop)) / max;

        int color = red << 16 | green << 8 | blue;

        return color;
    }
```

El método estático `paintGradient` recibe como parámetros el objeto gráfico que se va a utilizar para pintar, la definición de colores de inicio y fin, el punto de origen del relleno, identificado por coordenadas cartesianas (x, y), el ancho y el alto medidos en pixeles, y la orientación de la gradiente, determinada por dos constantes (`VERTICAL` y `HORIZONTAL`). Este método pinta el gradiente línea por línea.

El color de cada una de las líneas del gradiente es calculado por el método estático `midColor`, que se encarga de generar las variaciones de tono entre los colores especificados.

El prototipo que implementa esta funcionalidad presenta un menú que permite seleccionar el tipo de gradiente que se debe pintar en la pantalla del dispositivo.

Cuando se selecciona la opción “Vertical Gradient” del menú principal de la aplicación prototipo, se muestra una pantalla que degrada el color azul en blanco, de arriba hacia abajo. Si se selecciona la opción “Horizontal Gradient”, se muestra una pantalla en la que el color verde se torna negro, variando de izquierda a derecha.

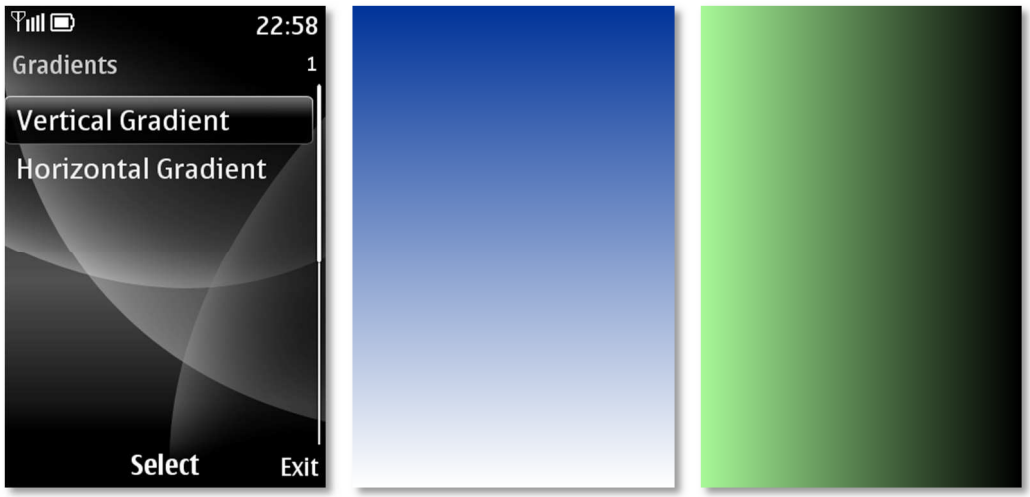



**Figura 2.3 Pantallas del prototipo de Pintado de Gradientes**
De izquierda a derecha: Menú Principal, Gradiente Vertical, Gradiente Horizontal
Elaborado por: Carlos Andrés Oquendo

**Inclusión y organización de componentes:** Determina la forma en que los componentes de la solución podrán interactuar con los contenedores.

La arquitectura de la solución debe permitir que el contenedor padre de una vista pueda incluir uno o más contenedores y/o componentes internos, creando una estructura jerárquica multinivel, de la que se puede ver una representación en la Figura 2.4.

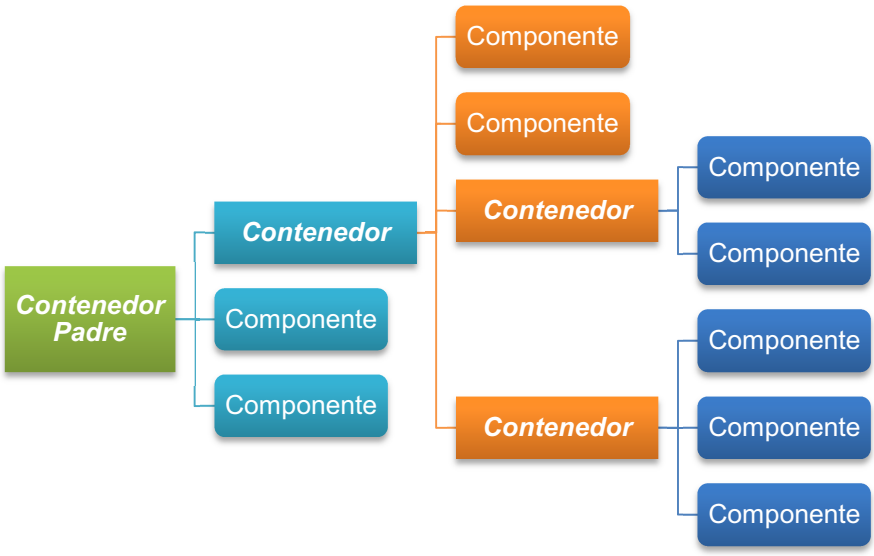



**Figura 2.4 Estructura jerárquica de inclusión de componentes.**
Elaborado por: Carlos Andrés Oquendo

Para que esta estructura funcione correctamente, es necesario que los

contenedores y los componentes tengan características que les permitan ser reconocidos como elementos de un mismo tipo.

Afortunadamente, los conceptos de herencia y polimorfismo de la programación orientada a objetos nos permiten satisfacer este requerimiento puntual.

El número de niveles de anidamiento de los contenedores podría ser ilimitado, siempre que se disponga de los recursos disponibles para soportarlo. También, un contenedor podría incluir un número ilimitado de componentes.

Las relaciones existentes entre contenedores y componentes pueden visualizarse en la Figura 2.5.

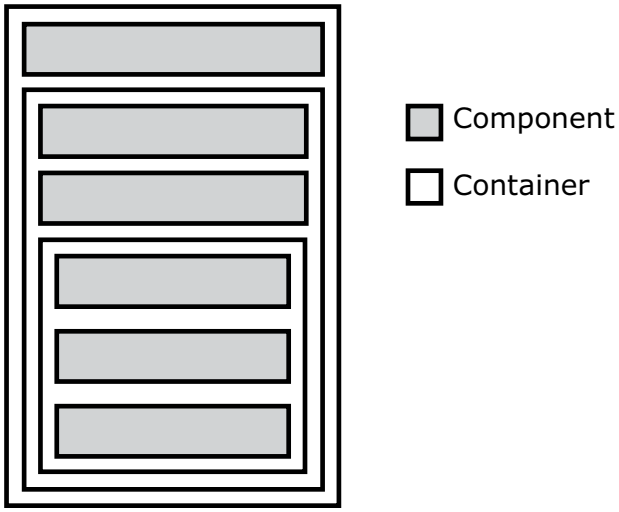



**Figura 2.5 Relaciones de inclusión entre Contenedores y Componentes**
Elaborado por: Carlos Andrés Oquendo

**Selección de Componentes:**

Permite establecer el estado de activación de uno de los componentes incluidos en una vista o contenedor.

Cuando un componente está seleccionado, puede interactuar y responder a los eventos de entrada generados por el usuario, al utilizar el teclado del dispositivo principalmente.

El contenedor principal debe mantener un indicador que le permita conocer que componente se debe seleccionar. Este indicador cambia cuando se detecta que el usuario presiona las teclas direccionales del dispositivo.

Todos los componentes seleccionables deben incluir un mecanismo que permita distinguirlos de aquellos que solo son capaces de mostrar información.

Esta funcionalidad se demostrará con un prototipo. Este implementará una lista de elementos que pueden ser seleccionables y, responderá a la actividad de las teclas de dirección superior e inferior.

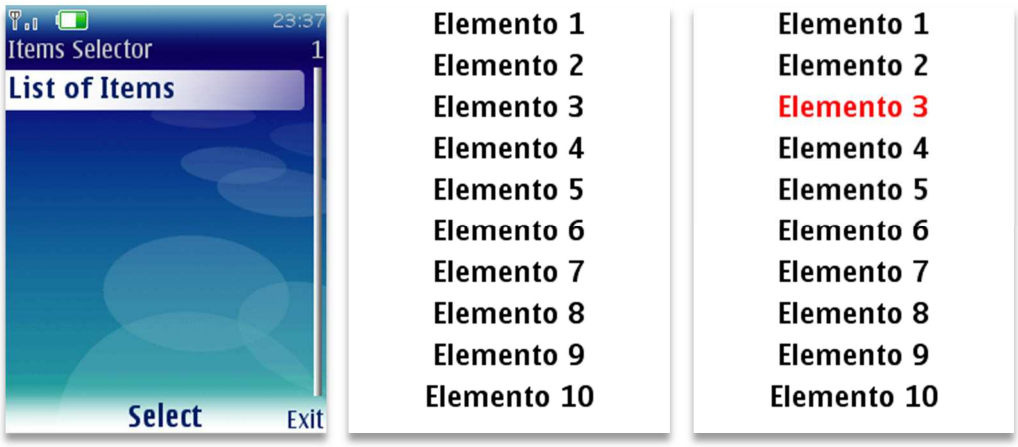



**Figura 2.6 Pantallas del Prototipo de Selección de Elementos**
De izquierda a derecha:
Menú Principal, Lista con elementos sin seleccionar, Lista con un elemento seleccionado.
Elaborado por: Carlos Andrés Oquendo

Cuando se selecciona la opción “List of Items” aparece una pantalla con una lista de elementos pintados de color negro. Cuando el usuario presiona indistintamente las teclas de dirección superior o inferior del dispositivo, el elemento de la lista que se encuentra en estado seleccionado cambia. Este comportamiento se evidencia cuando uno de los elementos de la lista se pinta de color rojo.

**Dimensionamiento:** El mecanismo de dimensionamiento permite determinar y establecer valores para el tamaño de un componente

respecto a su contenedor, en función del tamaño de la pantalla en el que se dibujen las interfaces gráficas de una aplicación.

Para que este mecanismo trabaje correctamente, es necesario que los tamaños de los componentes y de los contenedores sean calculados en forma dinámica, considerando algunos parámetros estándar que aplican para todos los elementos que se dibujan en la pantalla, como por ejemplo, los márgenes, el relleno, etc., basándose en la aplicación de un modelo de caja –similar al de Cascading Style Sheet (CSS, Hojas de Estilo en Cascada) que aplica para la tecnología HTML – que se puede revisar en la Figura 2.7.

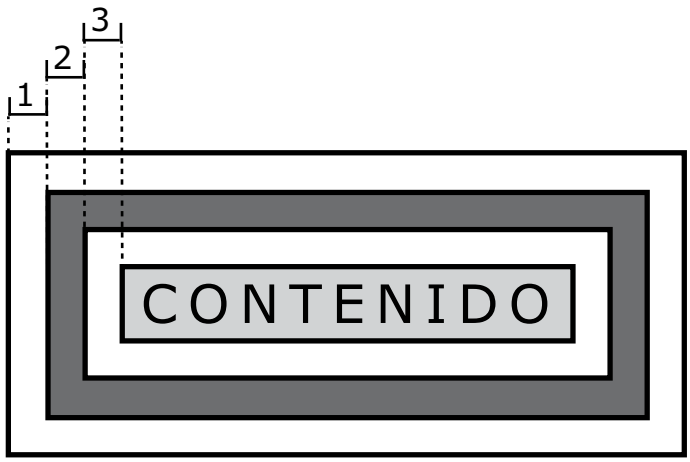



**Figura 2.7 Modelo de caja básico de Cascading Style Sheets (CSS)**
1. Ancho del margen externo
2. Ancho del borde, 3. Ancho del relleno (margen interno)
Elaborado por: Carlos Andrés Oquendo

Para la solución, se utilizará una simplificación de este modelo. Los contenedores se encargarán de manejar el margen externo y los controles de usuario dispondrán de un relleno (margen interno).

En general, los contenedores y los componentes adaptarán su altura dependiendo del contenido que posean. El ancho se deberá ajustar siempre al ancho disponible en la pantalla.

**Cambio de Vistas:** Este mecanismo se encarga de gestionar el modo en el que una vista, que se encuentra en estado visible, es reemplazada por otra vista diferente, ante la ejecución de un evento generado por el usuario

en la primera vista, o en vistas externas.

Se considerarán dos modos de cambio de vistas: administrado y libre.

En el modo administrado, el cambio de las vistas depende de la lógica de navegación especificada en un controlador de vistas. En el modo libre, una vista puede incluir llamadas a otras vistas desde los componentes internos que incluya y cada una de las vistas debe incluir componentes que le permitan realizar el cambio de vistas.

Para el modo administrado, se considerarán los siguientes tipos de controladores de vistas: basado en pestañas, y navegable.

El controlador basado en pestañas debe mostrar opciones que le permitan al usuario escoger la vista a la que se desea desplazar. El controlador navegable permite al usuario desplazarse entre vistas en un orden predefinido. Este controlador debe proporcionar al usuario opciones para regresar a una vista anterior.

Finalmente se procede a definir la arquitectura que permitirá realizar el desarrollo de la solución.

Aplicación Práctica: UIL 4 MIDP

En este punto se catalogará los componentes que se incorporarán a la solución:

| Categoría | Componente | Descripción |
|---|---|---|
| **Visualización** | Window | Es el componente que se vincula con el modelo de programación de Java ME. Se encarga de mostrar las vistas de una aplicación, a través de los controladores. |
| **Controladores** | Controller | Este es el componente que define la estructura básica de los controladores de vistas, que se encargan de administrar la forma en que las vistas se muestran en pantalla e interactúan con el usuario. |
| | StandaloneController | Controlador que gestiona y muestra en la pantalla una única vista a la vez. |
| | NavigableController | Controlador que tiene la capacidad de gestionar varias vistas y provee servicios de navegación automáticos. |

| | | |
|---|---|---|
| | TabsController | Controlador que administra un conjunto de vistas o controladores secundarios y que provee acceso a ellos a través de una barra de pestañas (con iconos) |
| **Vistas** | View | Componente genérico diseñado para mostrar información. |
| | Form | Vista básica para el diseño de las interfaces gráficas de usuario de una aplicación. Permite el uso *libre* de diferentes componentes visuales. |
| | List | Vista derivada de Form, que se enfoca en mostrar información en forma de listados de elementos. |
| | Dialog | Vista diseñada para interactuar con el usuario de manera exclusiva, y llamar su atención hacia un determinado evento. |
| **Componentes Visuales** | VisualComponent | Componente genérico que representa a cualquiera de los elementos visuales que pueden aparecer como parte de una vista. |
| | Container | Componente visual que es capaz de incorporar y organizar un conjunto de componentes visuales |
| | UserControl | Componente visual que posee una apariencia y una funcionalidad determinada. |
| **Contenedores** | StackedContainer | Contenedor que organiza un conjunto de varios componentes visuales de forma apilada. |
| | HorizontalSplittedContainer | Contenedor que organiza dos componentes visuales de forma horizontal, uno junto al otro. |
| **Controles de Usuario** | Button | Control que permite la ejecución de acciones respondiendo al evento de teclado adecuado. |
| | ImageBox | Control de usuario diseñado para mostrar imágenes en una vista. |
| | Label | Control diseñado para mostrar texto que no puede ser modificado directamente. |
| | ListItem | Control que es utilizado por List para mostrar los elementos de lista que contiene. Puede incluir texto e imágenes. |
| | OptionPicker | Control que administra un conjunto de valores y permite al usuario seleccionar uno de ellos. |
| | Switch | Control que permite manejar valores de tipo booleano (verdadero o falso). |
| | TextBox | Control diseñado para permitir al usuario ingresar información basada en texto. |

**Tabla 2.21. Catálogo de los Componentes de la Solución**
Elaborado por: Carlos Andrés Oquendo

Luego, es necesario representar la arquitectura de la solución y así poder observar las relaciones básicas que existen entre los componentes.

Para expresar la arquitectura básica de la solución, se utilizarán varias vistas de

las clases participantes. La vista general de las clases se incluye en la Figura 2.8.

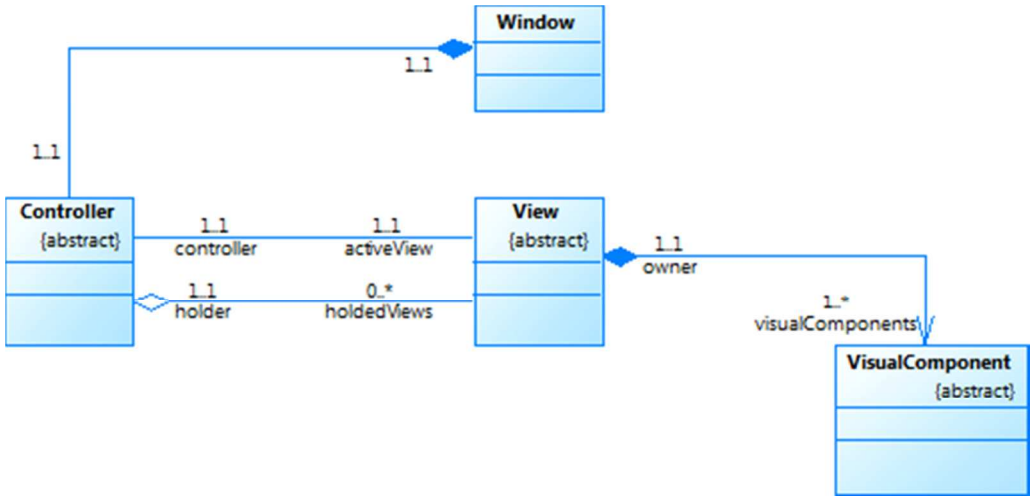



**Figura 2.8 Diagrama de Clases Principal de UIL 4 MIDP**
Elaborado por: Carlos Andrés Oquendo

El detalle de algunas de las clases de este diagrama, se podrá observar en esquemas adicionales.

En la Figura 2.9 se puede visualizar las tres diferentes especializaciones de la clase abstracta `Controller`: `TabsController`, `NavigableController` y `StandaloneController`.

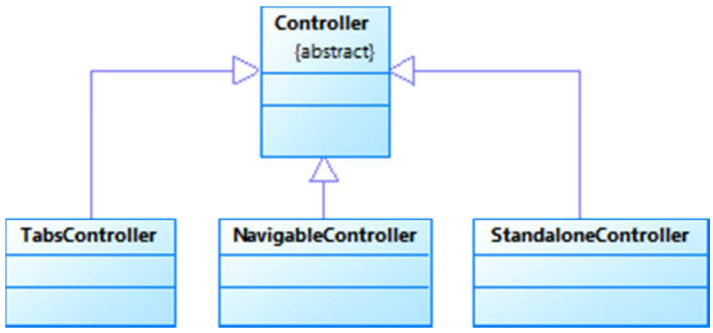



**Figura 2.9 Diagrama de Clases para la Clase Abstracta Controller**
Elaborado por: Carlos Andrés Oquendo

En la Figura 2.10 se aprecia las posibles especializaciones de la clase abstracta `View`: `Form` y `Dialog`. La primera debe permitir el diseño libre de pantallas, y la segunda debe orientarse a mostrar confirmaciones o notificaciones al usuario.

Además, la clase `List`, que es una especialización de `Form`, debe permitir y facilitar el manejo y visualización de información utilizando un diseño controlado.

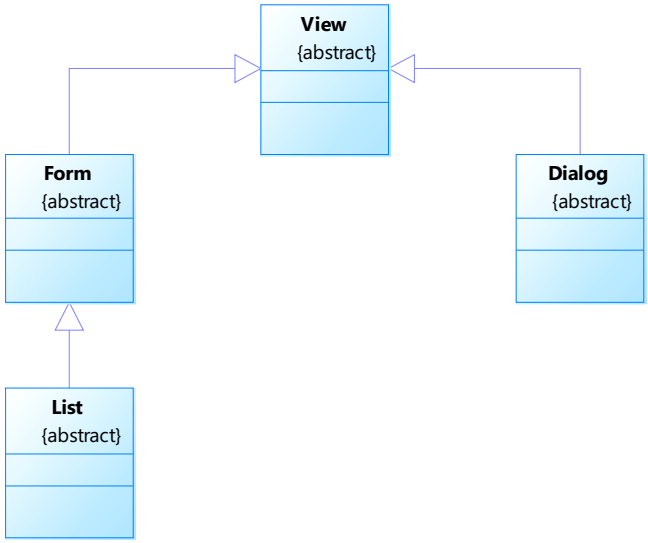



**Figura 2.10 Diagrama de Clases para la Clase Abstracta View**
Elaborado por: Carlos Andrés Oquendo

En la Figura 2.11 se muestra a las clases `Container` y `UserControl`, que heredan directamente la clase `VisualComponent`. Las posibles especializaciones de estas dos subclases se visualizan en las Figuras 2.12 y 2.13 respectivamente.

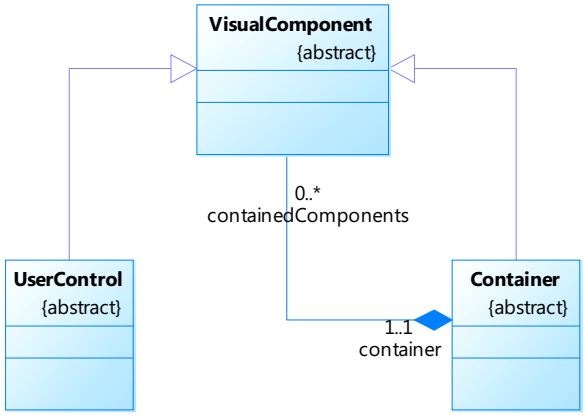



**Figura 2.11 Diagrama de Clases para la Clase Abstracta VisualComponent**
Elaborado por: Carlos Andrés Oquendo

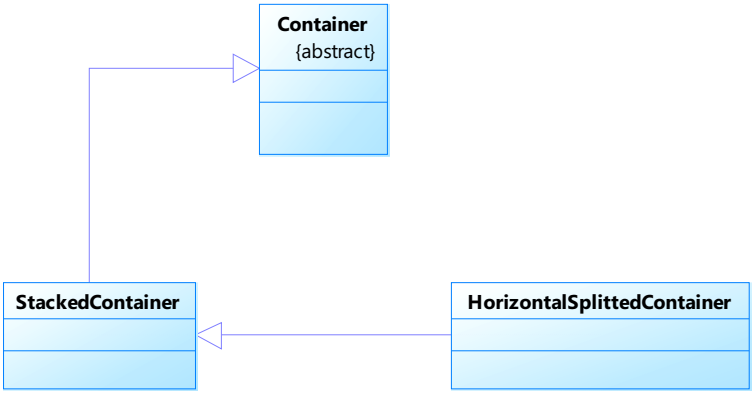



**Figura 2.12 Especializaciones de la Clase Abstracta Container**
Elaborado por: Carlos Andrés Oquendo

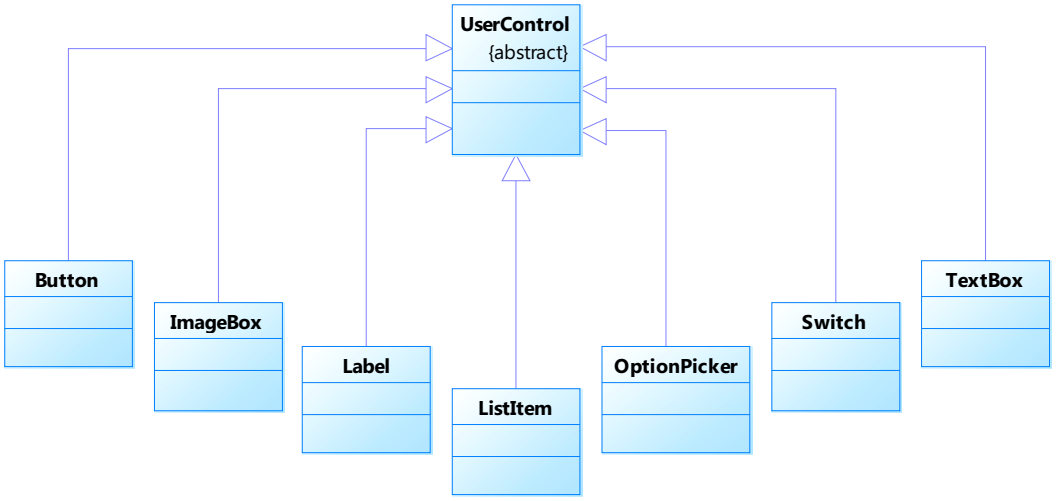



**Figura 2.13 Especializaciones de la clase UserControl**
Elaborado por: Carlos Andrés Oquendo

En todos los diagramas presentados únicamente se ha mostrado la relación entre las diferentes clases e interfaces que componen la solución.

Durante el avance del proyecto se desarrollarán todas las clases e interfaces. En el artefacto “Cuaderno de Arquitectura” se incluirán los diagramas completos.

### 2.2.4. DESARROLLAR INCREMENTO DE LA SOLUCIÓN

Esta tarea comprende el diseño de parte de la solución, para empezar a cumplir con un subconjunto de los requerimientos establecidos, funcionales o no, en escenarios y contextos específicos, como el acceso a la base de datos, diseños de interfaces gráficas de usuario, reglas de negocio, etc.

#### 2.2.4.1. DESARROLLO DE LA ACTIVIDAD

La primera tarea que se debe realizar, consiste en diseñar parte de la solución, en función de un subconjunto pequeño de los requerimientos, funcionales y no funcionales.

Aplicación Práctica: UIL 4 MIDP

Para la solución se realizará el diseño general de la apariencia de los controles y componentes que permiten la construcción de interfaces gráficas de usuario.

El diseño visual se realizará utilizando la herramienta Adobe Photoshop, que permite realizar el manejo de gráficos y colores de forma no destructiva, es decir,

aplicando estilos y capas que modifican la apariencia de los gráficos y que pueden ser incluidos o retirados sin alterar el gráfico base.

Adicionalmente, se utilizará la herramienta Adobe Kuler (http://kuler.adobe.com), que permite la creación y la utilización de esquemas de colores. Esta aplicación se encuentra disponible en línea, de manera gratuita.

Para asegurar que la apariencia proporcionada por la solución sea lo más acorde a las tendencias de diseño gráfico actual, se revisarán, como fuente de inspiración, algunas soluciones orientadas a la creación de aplicaciones móviles para Internet, como JQTouch (http://www.jqtouch.com) o jQuery Mobile Framework (http://www.jquerymobile.com), además de las interfaces gráficas de usuario de los sistemas operativos modernos para dispositivos móviles como Apple iOS 4, Google Android, Nokia Series 40 6th Edition Feature Pack 1, Symbian^3 y Microsoft Windows Phone 7.

El diseño propuesto se puede apreciar en las Figuras 2.14 y 2.15:

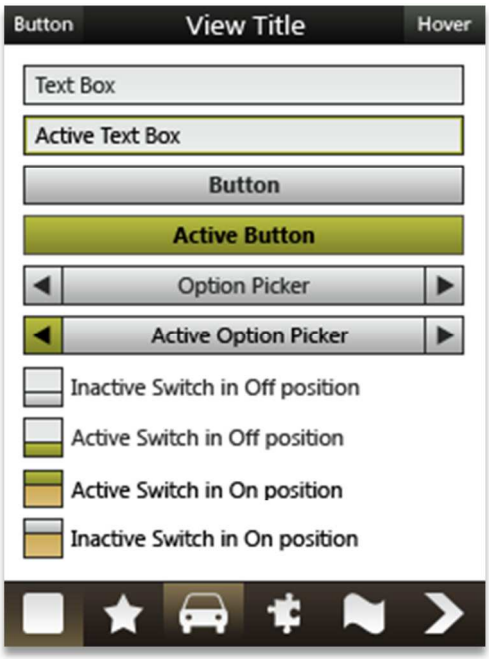



**Figura 2.14 Apariencia general de los controles de la solución**
Elaborado por: Carlos Andrés Oquendo

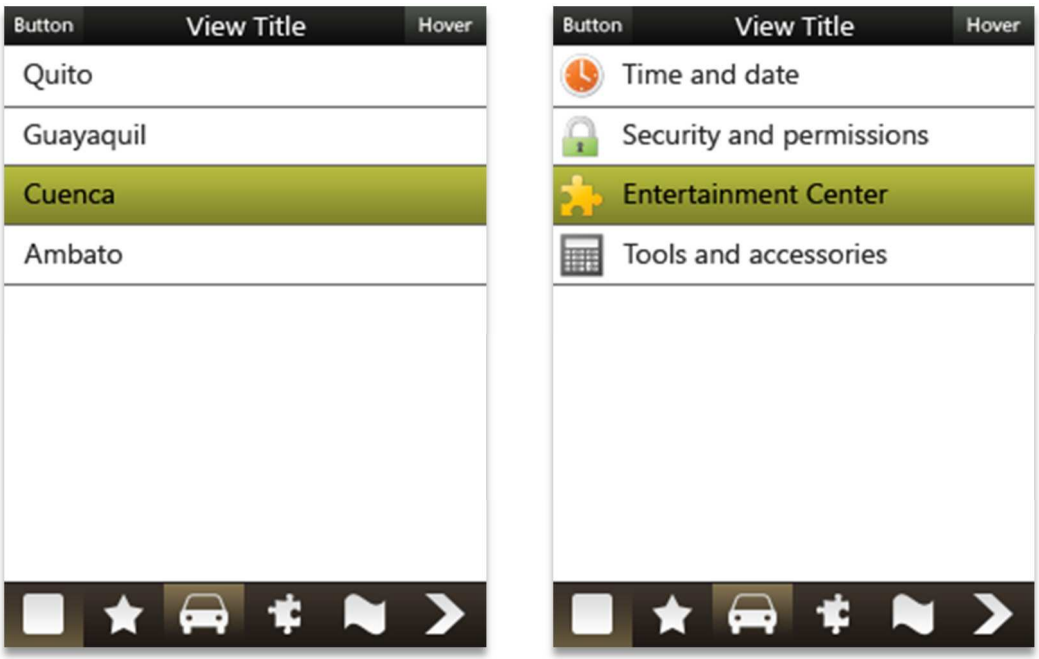



**Figura 2.15 Apariencia de las pantallas basadas en listas**
Elaborado por: Carlos Andrés Oquendo

La siguiente tarea comprende empezar a implementar la solución, enfocando el esfuerzo en un elemento específico, como una clase o un componente.

La metodología define un conjunto de pasos para lograr completar esta tarea. Se inicia verificando las posibilidades de reutilización de elementos y recursos. Luego, se puede generar una parte de código fuente que permita ir cumpliendo con el diseño propuesto. Durante estas actividades se podrá hacer cambios a algunos elementos de diseño para asegurar su consistencia con la implementación.

Aplicación Práctica: UIL 4 MIDP

En esta primera tarea de implementación se establece la estructura general de la biblioteca de clases. Se definirán los paquetes necesarios y se empezará a incorporar las clases requeridas, aunque su implementación no esté completa.

Este producto no podrá ser incluido en el cuerpo de este documento. Por esta razón, se reservará una copia del código fuente y se la incluirá junto a los demás artefactos. Esta copia del código fuente será marcada con la versión 0.50.

En la Figura 2.16 se puede ver la implementación de la estructura básica de paquetes de la solución como un proyecto dentro del entorno de desarrollo.

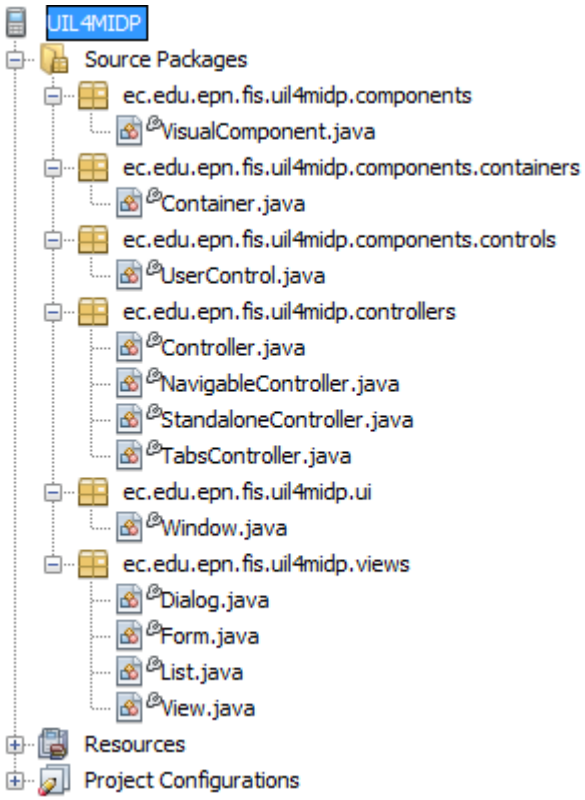



**Figura 2.16 Implementación de la estructura básica de paquetes de la solución**
Elaborado por: Carlos Andrés Oquendo

## 2.3. FASE DE CONSTRUCCIÓN

En la fase de construcción, el esfuerzo que se debe realizar se enfoca en completar el desarrollo del producto, implementando los requerimientos descritos y generando una versión estable que pueda ser usada directamente por los usuarios.

En el caso de la solución UIL4MIDP, el producto de esta fase será la biblioteca de clases completa, que luego podrá ser utilizada para crear una aplicación donde se aprecie su funcionalidad.

### 2.3.1. PLANEAR Y GESTIONAR LAS ITERACIONES

Durante esta fase del proyecto se realiza la asignación de elementos de trabajo a cada uno de los miembros del equipo. Cada elemento de trabajo debe ser descompuesto en actividades más pequeñas de tal manera que se pueda realizar una estimación realista del esfuerzo y tiempo que se necesita para llevarlas a cabo.

Al finalizar la actividad, se deberá actualizar la Lista de Elementos de Trabajo, y si es necesario, la Lista de Riesgos del proyecto.

#### 2.3.1.1. DESARROLLO DE LA ACTIVIDAD

La actividad inicia haciendo una revisión de la lista de elementos de trabajo existente de la Fase de Elaboración. Cada elemento de trabajo será desglosado, priorizado y se asignarán tiempos específicos para las tareas resultantes.

Aplicación Práctica: UIL 4 MIDP

Los Elementos de Trabajo que todavía no han sido completados se desglosan en la Tabla 2.22, que aparece a continuación:

| **Tarea** | **Tiempo Estimado (Horas)** |
|---|---|
| Validar el diseño de la interfaz gráfica | 8 |
| Establecer el conjunto de componentes a construir. | 8 |
| **Construir y documentar un subconjunto básico de componentes.** | **96** |
| *Construir Window* | *8* |
| *Construir Controller y StandaloneController* | *12* |
| *Construir View* | *8* |
| *Construir VisualComponent* | *8* |
| *Construir Container y StackedContainer* | *12* |
| *Construir UserControl* | *8* |
| *Construir Label* | *8* |
| *Construir TextBox* | *10* |
| *Construir Button* | *10* |
| *Construir Form* | *12* |
| Construir aplicaciones de prueba con los componentes básicos. | 24 |
| **Construir y documentar los componentes restantes.** | **80** |
| *Construir OptionPicker* | *6* |
| *Construir Switch* | *6* |
| *Construir ImageBox* | *8* |
| *Construir ListItem* | *6* |
| *Construir List* | *8* |
| *Construir HorizontalSplittedContainer* | *6* |

| | |
|---|---|
| *Construir TabsController* | *10* |
| *Construir NavigableController* | *10* |
| *Construir Dialog* | *12* |
| *Construir Otros Componentes* | *8* |
| Construir aplicaciones de prueba de los componentes restantes | 32 |
| Investigar el funcionamiento de mecanismos y elementos desconocidos | 16 |

**Tabla 2.22. Desglose de las Tareas de Construcción**
Elaborado por: Carlos Andrés Oquendo

Estas tareas se incorporarán al artefacto Lista de Elementos de Trabajo, donde se establecerá el estado de asignación de la tarea y el responsable de ejecutarla.

Si se aprecia la aparición de riesgos adicionales a los ya establecidos en el entregable denominado Lista de Riesgos, es necesario incorporarlos, detallarlos y valorarlos en el entregable mencionado.

Aplicación Práctica: UIL 4 MIDP

Para el caso de la solución, la revisión de la lista de elementos de trabajo no ha visibilizado riesgos adicionales, por lo que no es necesario actualizar por el momento el artefacto asociado.

### 2.3.2. IDENTIFICAR Y REFINAR LOS REQUERIMIENTOS

El enfoque de esta actividad nuevamente cambia y se centra en asegurarse que los requerimientos para la solución se encuentran estables y que el nivel de detalle de éstos es suficiente y permite comprender con facilidad su alcance y funcionalidad.

Esta actividad se realiza tomando como entradas los artefactos generados en la fase de elaboración, específicamente, el modelo y las descripciones de casos de uso, el documento de requerimientos globales, y el glosario.

#### 2.3.2.1. DESARROLLO DE LA ACTIVIDAD

Esta actividad comienza con la revisión del artefacto de requerimientos globales. Si aparecen requerimientos no contemplados, se debe añadirlos al artefacto.

Es también prudente revisar el detalle del modelo de casos de uso y las descripciones asociadas, para asegurarse que se encuentran consistentes con los requerimientos del sistema.

Aplicación Práctica: UIL 4 MIDP

Se ha realizado una revisión de los requerimientos globales del sistema y del modelo de casos de uso y sus descripciones. Al finalizar la revisión, se ha determinado que no hay requerimientos faltantes y que los casos de uso y sus descripciones se encuentran correctos. Por estos motivos, no es necesario actualizar los artefactos descritos.

### 2.3.3. DESARROLLAR INCREMENTO DE LA SOLUCIÓN

Esta actividad se enfoca en llevar a cabo los elementos de trabajo o tareas descritos para el proyecto.

Para tener éxito en la implementación de la solución, se debe asegurar que el diseño se encuentra completo y estable, para cada uno de los componentes involucrados. Adicionalmente, puede ser necesario ir ejecutando pruebas de unidad para asegurar el correcto funcionamiento de los componentes.

#### 2.3.3.1. DESARROLLO DE LA ACTIVIDAD

La primera tarea que se debe realizar en esta actividad es el diseño de los componentes de la solución. El diseño se puede expresar de manera visual, utilizando diagramas y gráficos, de forma que resulte más sencilla su comprensión y aplicación en la construcción de la solución.

Con el diseño terminado se procede a realizar la implementación del producto de software, escribiendo el código fuente de los componentes según se indique en el diseño elaborado. A medida que se implementa la solución, puede ser necesario ir haciendo pruebas para asegurar el correcto funcionamiento de los componentes de la solución.

Cuando la programación ha terminado, se integra los componentes y se crea una versión ejecutable o compilada, que se encuentra lista para ser utilizada.

Aplicación Práctica: UIL 4 MIDP

La primera actividad que debe ser efectuada es revisar el conjunto de componentes que se va a construir.

De la revisión realizada, se ha determinado que, además de los componentes de propósito general establecidos en las fases anteriores, es necesario construir los siguientes para poder completar la funcionalidad requerida:

| Categoría | Nombre del Componente | Descripción |
|---|---|---|
| **Vistas** | MessageDialog | Este tipo de diálogo permite mostrar al usuario un mensaje que únicamente puede ser aceptado. Para esto, se provee un botón que acepta y permite cerrar el mensaje. |
| | ConfirmationDialog | Este diálogo presenta al usuario un mensaje y opciones para permitir al usuario emitir una respuesta afirmativa o negativa respecto a la sugerencia indicada en pantalla. Ambas opciones permiten cerrar el mensaje. |
| | ProgressDialog | Esta vista de diálogo permite mostrar una indicación donde se aprecia que alguna actividad se está realizando. No incluye opciones utilizables por el usuario para cerrar el mensaje, sino que el proceso que invoca el diálogo debe cerrarlo. |
| | SplashScreen | Es una vista especial que facilita la inclusión de pantallas de inicio en las que se muestran marcas o información y se podrían ejecutar tareas adicionales de inicialización de la aplicación. |
| **Controles de Usuario** | AnimatedImageBox | Es un control de usuario que se encarga de automatizar la visualización de animaciones pequeñas. Este control es principalmente para uso interno de la solución. |
| | TabBar | Es un control de usuario que incorpora un conjunto de pestañas. Cada pestaña se asocia a una vista. Este control se utiliza únicamente por TabsController y tiene la capacidad de permitir al usuario acceder a una de las vistas relacionadas al controlador. |
| | Tab | Es un control de usuario que representa a una pestaña. Muestra siempre un icono. Es utilizado únicamente por TabBar. |
| | TitleBar | Es un control de usuario que representa a la barra de título de una vista. Puede incorporar hasta dos botones y un texto explicativo. Este control se utiliza únicamente por las vistas: Form, List, Dialog, etc. |
| | TitleBarButton | Ccontrol de usuario que permite ejecutar acciones. Es capaz de mostrar texto o iconos. Solamente debe ser utilizado por TitleBar. |
| **Utilitarios** | FontManager | Es un componente que facilita la utilización de fuentes en los diferentes componentes de la solución |
| | FramesManager | Es un componente que se encarga de administrar las imágenes que forman parte de una animación. Es utilizado principalmente por AnimatedImageBox |
| | GradientManager | Este componente tiene toda la lógica necesaria para realizar el pintado de gradientes. |

| | | |
|---|---|---|
| | ResourceManager | Este componente gestiona los recursos de imagen que se utilizan en la aplicación. |
| | TextManager | Este componente incluye métodos que facilitan la manipulación de texto. |
| | ThemeManager | Gestiona el uso de colores, recursos y dimensiones para todos los componentes que conforman la solución. |

**Tabla 2.23. Componentes Adicionales para la Solución**
Elaborado por: Carlos Andrés Oquendo

Una vez que ya se tiene una definición más certera y completa de los componentes que se debe desarrollar, se puede pasar a completar el diseño de la solución, que se inició en la fase de elaboración.

La complementación del diseño existente comprenderá:

- Incorporar descripciones de las clases necesarias.
- Incluir en los diagramas de clases los elementos necesarios para hacer frente a los requerimientos.
- Completar la estructura interna de las clases con atributos y métodos.
- Crear diagramas de clases adicionales auxiliares, donde se representen relaciones de valor entre las diferentes clases que componen la solución.

El diseño se efectuará utilizando la herramienta Sybase PowerDesigner, principalmente, por la flexibilidad que ofrece para la elaboración de diagramas.

Para facilitar la visualización y comprensión del diseño, se incluye como parte de los entregables un reporte basado en el modelo donde se aprecia toda la estructura y detalles del diseño.

En las Figuras 2.17 y 2.18 se aprecia ejemplos de los diagramas de clases utilizados como parte del diseño de los componentes de la solución.

En la Figura 2.19 se incluye una imagen donde se aprecia la estructura general del modelo para la solución dentro de la herramienta de diseño.

Una vez concluido el diseño, es necesario pasar a la implementación en sí de la solución. Para el caso de la solución UIL4MIDP se realizará la implementación de los componentes según el diseño elaborado, utilizando como punto de inicio el proyecto definido en la fase de elaboración.

Como parte de la construcción, se incorporarán paquetes adicionales faltantes,

se crearán clases utilitarias, se añadirán archivos de recursos y se escribirá el código que corresponde a cada uno de los componentes de la solución.

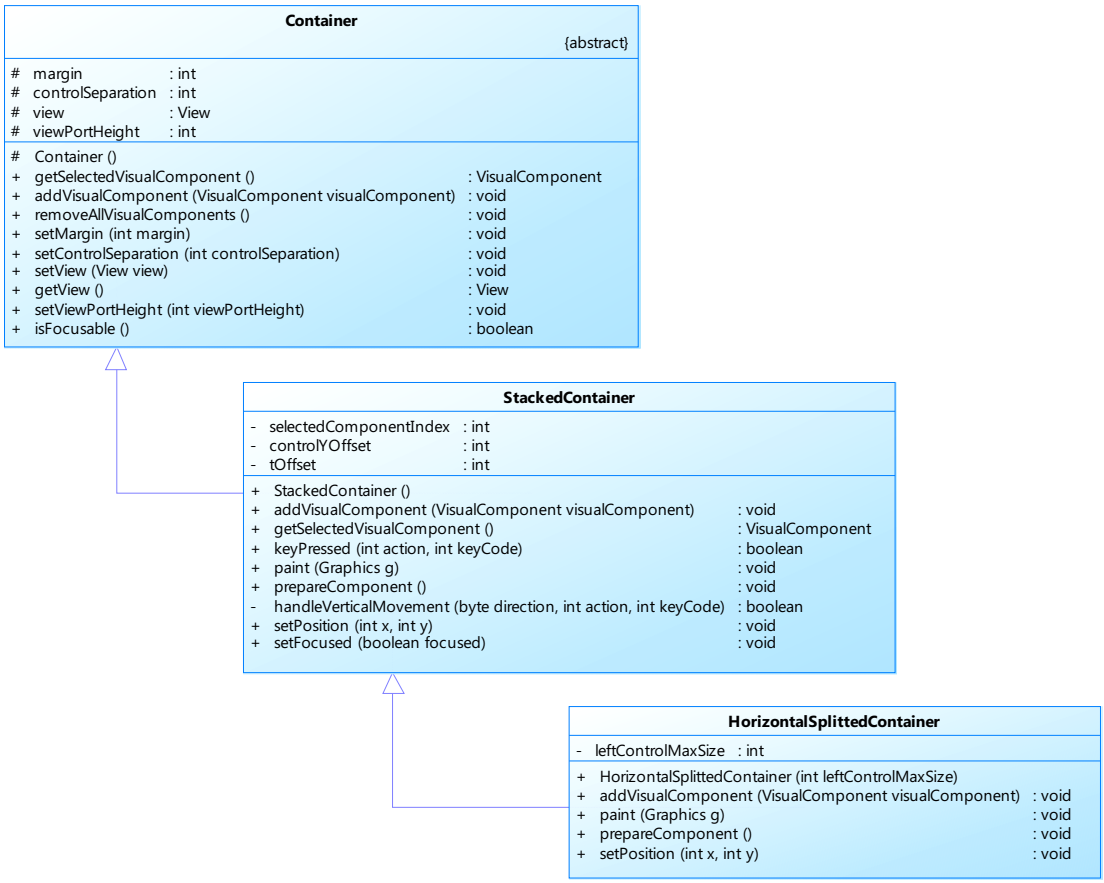



**Figura 2.17. Diagrama de Clases General de los Contenedores**
Elaborado por: Carlos Andrés Oquendo

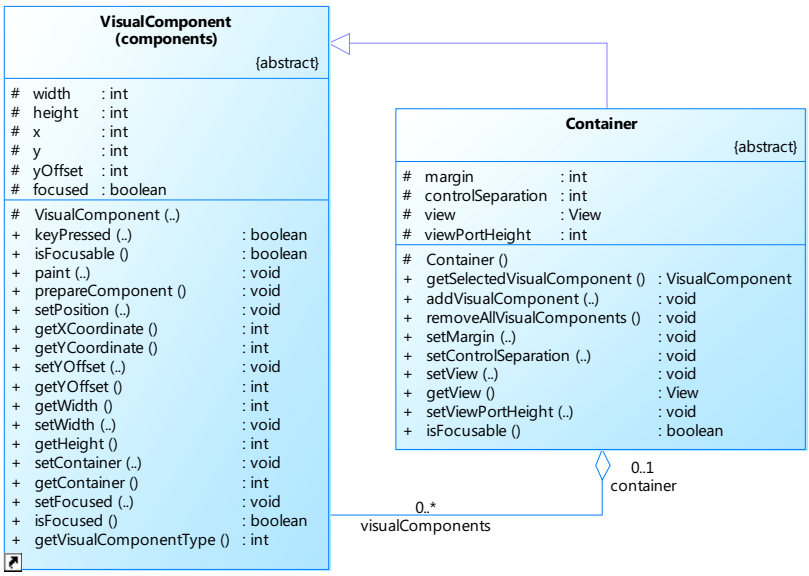



**Figura 2.18. Diagrama de Clases Auxiliar: Container – VisualComponent**
Elaborado por: Carlos Andrés Oquendo

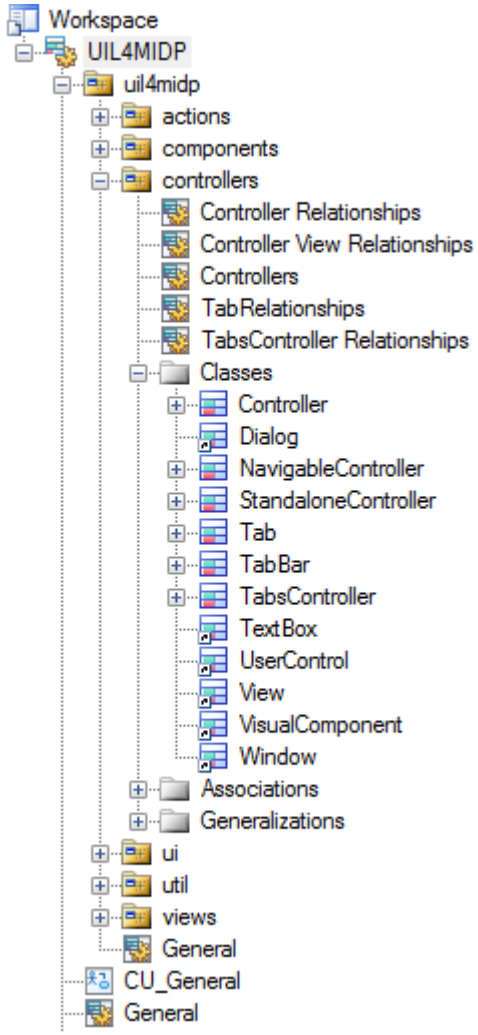



**Figura 2.19. Ejemplo de la Estructura del Diseño en PowerDesigner**
Elaborado por: Carlos Andrés Oquendo

En el proceso de la escritura del código fuente se procurará:

- Seguir las reglas de sintaxis definidas para el lenguaje de programación Java.
- Escribir comentarios que formarán parte de la documentación de programación Javadoc.
- Mantener organizado el código utilizando las facilidades que brinda el entorno integrado de desarrollo NetBeans IDE.

En el código fuente que aparece a continuación, se aprecia un ejemplo de la programación realizada para la clase Label, que es uno de los componentes visuales que conforman la solución.

```
package ec.edu.epn.fis.uil4midp.components.controls;

import ec.edu.epn.fis.uil4midp.util.FontManager;
import ec.edu.epn.fis.uil4midp.util.TextManager;
import javax.microedition.lcdui.Graphics;

/**
 * A Label is a VisualComponent intended to display any text values.
 * @author Carlos Andrés Oquendo
 */
public class Label extends UserControl {

    public static final int LABEL_LEFT = Graphics.LEFT;
    public static final int LABEL_CENTER = Graphics.HCENTER;
    public static final int LABEL_RIGHT = Graphics.RIGHT;
    private String caption = "";
    private String[] captionLines;
    private boolean captionSynced;
    private int textAlignment = LABEL_LEFT;

    //<editor-fold desc="Constructors">
    /**
     * Creates a new Label instance without a text as caption.
     */
    public Label() {
        font = FontManager.getNormalFont();
        captionSynced = false;
    }

    /**
     * Creates a new Label instance with the specified caption.
     * @param caption Text to show in the Label.
     */
    public Label(String caption) {
        this();
        this.caption = caption;
    }
    //</editor-fold>

    //<editor-fold desc="Getters & Setters">
    /**
     * Gets the caption of the Label
     * @return String containing the caption of the Label
     */
    public String getCaption() {
        return caption;
    }

    /**
     * Sets the caption of the Label
     * @param caption Caption of the Label
     */
    public void setCaption(String caption) {
        this.caption = caption;
        captionSynced = false;
    }

    /**
     * Sets the alignment of the text on the label
     * @param textAlignment Alignment of the text on the label. The available values
     * are Label.LABEL_LEFT, Label.LABEL_CENTER and Label.LABEL_RIGHT.
     */
```

```
    public void setTextAlignment(int textAlignment) {
        this.textAlignment = textAlignment;
    }
    //</editor-fold>

    //<editor-fold desc="Abstract Methods Implementations">
    /**
     * Handles the key events.
     * Label can not handle any key press event.
     * @param action Canvas' key action number.
     * @param keyCode Pressed key code. This code may be device-specific.
     * @return This method always returns False.
     */
    public boolean keyPressed(int action, int keyCode) {
        return false;
    }

    /**
     * Determines if the Label can be focused.
     * @return This method always return False, due to Label can not be focused.
     */
    public boolean isFocusable() {
        return false;
    }

    /**
     * Paints the Label.
     * @param g Graphics object on which paint.
     */
    public void paint(Graphics g) {
        prepareComponent();

        int ty = y - yOffset;

        // Draw text
        g.setColor(tm.getPrimaryFontColor());
        g.setFont(font);

        int[] pos = new int[]{0, ty + padding};
        switch (textAlignment) {
            case LABEL_LEFT:
                pos[0] = x + padding;
                break;
            case LABEL_CENTER:
                pos[0] = width / 2;
                break;
            case LABEL_RIGHT:
                pos[0] = width - padding;
                break;
        }

        for (int i = 0; i < captionLines.length; i++) {
            g.drawString(captionLines[i],
                pos[0], pos[1] + font.getHeight() * i, Graphics.TOP | textAlignment);
        }
    }

    /**
     * Prepares the layout of the Label.
     */
    public void prepareComponent() {
        if (!captionSynced || !layoutSynced) {
            int captionLineWidth = width - 2 * padding;
```

```
            captionLines = TextManager.getLines(caption, captionLineWidth, font);
            height = captionLines.length * font.getHeight() + 2 * padding;

            captionSynced = true;
            layoutSynced = true;
        }
    }
    //</editor-fold>
}
```

Una vez concluida la programación, se compilará una versión de la solución UIL4MIDP. En esta versión de la librería se incluirán los siguientes entregables:

- Archivos Compilados:
    - UIL4MIDP.jad – Descriptor de la biblioteca de clases.
    - UIL4MIDP.jar – Archivo que contiene la biblioteca de clases (para desarrollo)
    - UIL4MIDP.min.jad – Descriptor de la biblioteca de clases en formato miniminado.
    - UIL4MIDP.min.jar – Archivo que contiene la biblioteca de clases en formato minimizado (para redistribuir con las aplicaciones)
- Documentación *Javadoc*

  Documentación de programación de la biblioteca de clases. Explica el uso de los métodos que conforman las diferentes clases así como de los atributos implementados en cada una de ellas.
- Código Fuente

  Incluye todo el código fuente de la solución así como el proyecto configurado para el entorno de desarrollo NetBeans IDE.

Para complementar el desarrollo de la solución UIL4MIDP, se debe escribir un manual de programación, en el que se explique de manera sencilla y ejemplificada, la forma en que se debe utilizar la biblioteca de clases para construir otras aplicaciones basadas en ella.

El Manual de Programación se incluirá como parte de los entregables de la solución.

Al terminar esta fase del proceso de desarrollo, se dispone de un producto de software funcional, que ya puede ser utilizado independientemente, y cuenta con suficiente documentación de su funcionamiento y características.

Tamaño de la Librería Compilada

Debido a que la solución UIL4MIDP es un marco de trabajo sobre el cual se pueden desarrollar aplicaciones, es necesario que su tamaño sea el mínimo posible.

Cuando se compila la solución UIL4MIDP usando las propiedades predeterminadas del entorno de desarrollo, es decir, sin ninguna opción de ofuscación y optimización activada, la biblioteca de clases se genera con un tamaño de aproximadamente 85 KB.

Esta versión de la biblioteca de clases (UIL4MIDP.jar) es adecuada para el desarrollo, ya que contiene las definiciones completas de los miembros que componen cada una de las clases, sus métodos y atributos.

Al cambiar las configuraciones de ofuscación y establecerlas en una modalidad adecuada para bibliotecas de clases, dónde principalmente se reduce la longitud de los nombres de las variables, clases y métodos privados, *default* y protegidos, y donde se aplican un conjunto de optimizaciones, la biblioteca de clases de la solución UIL4MIDP se genera con un tamaño de aproximadamente 67 KB.

Esta versión (UIL4MIDP.min.jar) adquiere características que la hacen más adecuada para redistribuirse conjuntamente con las aplicaciones que la utilicen. Eventualmente, y con el uso de herramientas adicionales de compresión y optimización, es posible que el tamaño de la biblioteca de clases pueda reducirse más, y de esta manera, su impacto en el tamaño final de una aplicación, se minimice.

En la Figura 2.20 se aprecia las diferencias de tamaño entre ambas versiones de la biblioteca de clases.

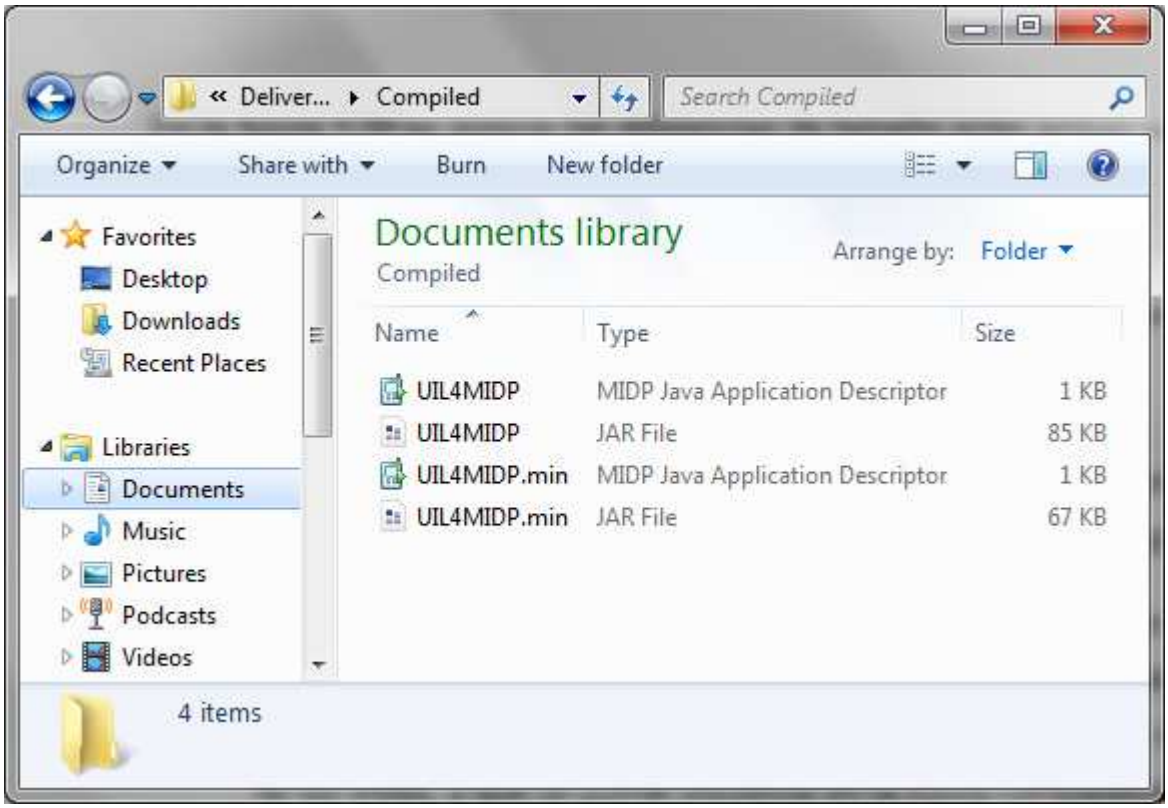



**Figura 2.20. Tamaños de los compilados de la Biblioteca de Clases UIL4MIDP**
Elaborado por: Carlos Andrés Oquendo

# CAPÍTULO 3. CREACIÓN DE UN PROTOTIPO COMO CASO DE ESTUDIO

Este capítulo se centrará en el desarrollo de una aplicación que utilice como base a la biblioteca de clases *User Interface Library for MIDP – UIL4MIDP*, que fue diseñada y desarrollada según lo que se encuentra documentado en el capítulo anterior.

Este prototipo servirá para verificar la compatibilidad y el funcionamiento de la biblioteca de clases en un conjunto de dispositivos móviles basados en la plataforma Java Micro Edition.

El diseño del prototipo se realizará utilizando los artefactos estrictamente necesarios para expresar sus características y funcionalidad. Es necesario indicar que el desarrollo del prototipo no es el aspecto central de la solución UIL4MIDP, sino un mecanismo que permite observarla en funcionamiento, y además ejemplificar la forma en que los desarrolladores de aplicaciones pueden utilizarla.

## 3.1. DEFINICIÓN DE REQUISITOS DEL PROTOTIPO

La aplicación que se desarrolle como prototipo permitirá observar en funcionamiento a los componentes desarrollados como parte de la solución UIL4MIDP.

Por este motivo, se ha determinado conveniente crear una aplicación cliente del servicio Twitter (http://www.twitter.com), que facilita la publicación y el seguimiento de mensajes cortos (*microblogging*).

Los justificativos principales, funcionales y técnicos, que influyeron en la decisión de construir el prototipo basado en el uso del servicio Twitter son los siguientes:

- Twitter es un servicio actual que tiene un número creciente de usuarios.
- Twitter se encuentra disponible en línea a nivel mundial, y puede ser accedido libremente.
- La funcionalidad que proporciona el servicio es muy definida.
- Twitter ofrece un API que puede ser consumida desde casi cualquier tecnología. Se requiere únicamente de un nombre de usuario y una contraseña del servicio para acceder al API y a la documentación asociada disponible en http://dev.twitter.com.

### 3.1.1. DESCRIPCIÓN FUNCIONAL

Twitter es un servicio de publicación de mensajes cortos, también conocido como *microblogging*. Para utilizar el servicio, los usuarios deben registrarse para obtener una cuenta de acceso. Con ella, es posible publicar información, encontrar y seguir a personas, suscribirse a sus publicaciones y visualizarlas tanto de forma independiente como agrupada. En la Figura 3.1 se puede observar la página principal del Cliente Web oficial del servicio.

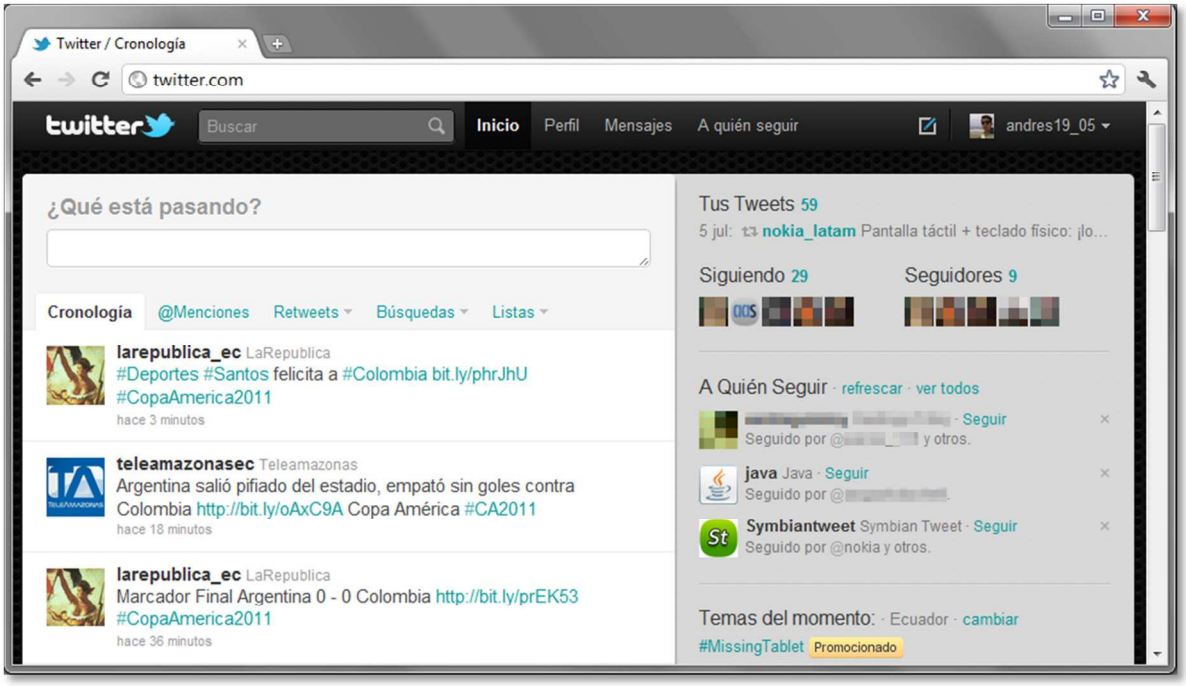



**Figura 3.1. Cliente Web del Servicio Twitter**
Elaborado por: Carlos Andrés Oquendo

### 3.1.2. REQUERIMIENTOS FUNCIONALES

El cliente del servicio Twitter basado en la biblioteca de clases UIL4MIDP se denominará *Tweet 4 ME*. Esta aplicación deberá permitir al usuario interactuar con el servicio de Twitter y realizar el siguiente conjunto de actividades generales:

- Ingresar al servicio Twitter.
- Mostrar la información personal del usuario y de otros usuarios.
- Mostrar las publicaciones del usuario y las menciones hechas a su perfil.
- Mostrar las publicaciones más recientes, o las de una persona específica.
- Ver el detalle de una publicación, publicar un mensaje nuevo, responder a una publicación o republicarla.

- Ver las personas que siguen y a las que sigue el usuario.
- Seguir a sus seguidores y a otras personas o dejar de seguir a una persona.
- Bloquear a un seguidor.

### 3.1.3. SELECCIÓN DE DISPOSITIVOS DE PRUEBA

La solución User Interface Library for MIDP – UIL4MIDP se desarrolló tomando en cuenta las características y funcionalidades disponibles en la plataforma Java Micro Edition, específicamente, el perfil MIDP 2.0 y la configuración CLDC 1.0 o superiores.

Con el fin de probar el funcionamiento de la solución UIL4MIDP, el prototipo Tweet 4 ME debe funcionar en la misma plataforma (perfiles y configuraciones).

Los dispositivos que se han seleccionado incluyen un entorno apropiado para probar la solución, y representan a una determinada plataforma operativa.

| No. | Dispositivo | Sistema Operativo | Versión de MIDP | Versión de CLDC | Tamaño de Pantalla |
|---|---|---|---|---|---|
| 1 | Nokia 6300 | Series 40 3rd Edition, Feature Pack 2 | 2.0 | 1.1 | 240 x 320 px. |
| | http://www.developer.nokia.com/Devices/Device_specifications/6300/ | | | | |
| 2 | Nokia 5530 XpressMusic | Series 60 5th Edition, Symbian OS 9.4 | 2.1 | 1.1 | 360 x 640 px. |
| | http://www.developer.nokia.com/Devices/Device_specifications/5530_XpressMusic/ | | | | |
| 3 | Nokia N8-00 | Symbian^3 | 2.1 | 1.1 | 360 x 640 px. |
| | http://www.developer.nokia.com/Devices/Device_specifications/N8-00/ | | | | |
| 4 | Sony Ericsson W302 | *No disponible* | 2.0 | 1.1 | 176 x 220 px. |
| | http://developer.sonyericsson.com/wportal/devworld/phones/phone-overview/w302?cc=gb&lc=en | | | | |
| 5 | Sony Ericsson W200 | *No disponible* | 2.0 | 1.1 | 128 x 160 px. |
| | http://developer.sonyericsson.com/wportal/devworld/phones/phone-overview/w200a?cc=gb&lc=en | | | | |
| 6 | Samsung GT M2310 | *No disponible* | 2.0 | 1.0 | 128 x 160 px. |
| | http://www.samsung.com/latin/consumer/mobile-devices/mobile-phones/archives/GT-M2310BBAJDI/index.idx?pagetype=prd_detail&tab=specification | | | | |
| 7 | Samsung GT E2121L | *No disponible* | 2.0 | 1.0 (N/D) | 128 x 128 px. |
| | http://www.samsung.com/latin/consumer/mobile-phones/mobile-phones/bar-phones/GT-E2121ARLECO/index.idx?pagetype=prd_detail&tab=specification | | | | |
| 8 | Samsung GT B3410 | *No disponible* | 2.0 | 1.1 | 240 x 320 px. |
| | http://www.samsung.com/latin/consumer/mobile-phones/mobile-phones/slider/GT-B3410JGLTTT/index.idx?pagetype=prd_detail&tab=specification | | | | |
| 9 | Motorola MOTOROKR U9 | MOTOMAGX | 2.0 | 1.1 | 240 x 320 px. |
| | http://developer.motorola.com/products/handsets-other/motou9/ | | | | |

**Tabla 3.1. Especificaciones de los Dispositivos de Prueba**
Elaborado por: Carlos Andrés Oquendo

## 3.2. ELABORACIÓN DEL PROTOTIPO

Durante esta actividad se realizará el diseño y la implementación del prototipo Tweet 4 ME.

Para ello se utilizará un conjunto de artefactos esenciales que permitan describir y comprender correctamente las funcionalidades que éste debe poseer.

### 3.2.1. MODELO DE CASOS DE USO DEL PROTOTIPO

A partir de la definición de las funcionalidades que deben ser incorporadas en el prototipo, se puede establecer los casos de uso que se encuentran enumerados en la Tabla 3.2.

| No. | Caso de Uso |
|---|---|
| 1 | Iniciar Sesión en Twitter |
| 2 | Ver información personal |
| 3 | Ver las publicaciones realizadas por uno mismo |
| 4 | Ver las menciones de mi perfil |
| 5 | Ver las personas a quienes sigo |
| 6 | Ver las personas que me siguen |
| 7 | Seguir a personas desconocidas |
| 8 | Ver Información de una persona |
| 9 | Seguir a uno de los seguidores |
| 10 | Dejar de seguir a una persona |
| 11 | Bloquear a un seguidor |
| 12 | Ver un listado de los tweets más recientes |
| 13 | Ver los tweets de una persona determinada |
| 14 | Visualizar el detalle de un tweet |
| 15 | Publicar un tweet nuevo |
| 16 | Retwittear una publicación de terceras personas |
| 17 | Responder a una publicación |

**Tabla 3.2. Casos de Uso de Tweet 4 ME**
Elaborado por: Carlos Andrés Oquendo

Es importante destacar que todos estos casos de uso se llevan a cabo por el usuario del servicio Twitter.

En la Figura 3.2 que se encuentra a continuación, se puede visualizar una representación de la relación existente entre el actor y los casos de uso definidos.

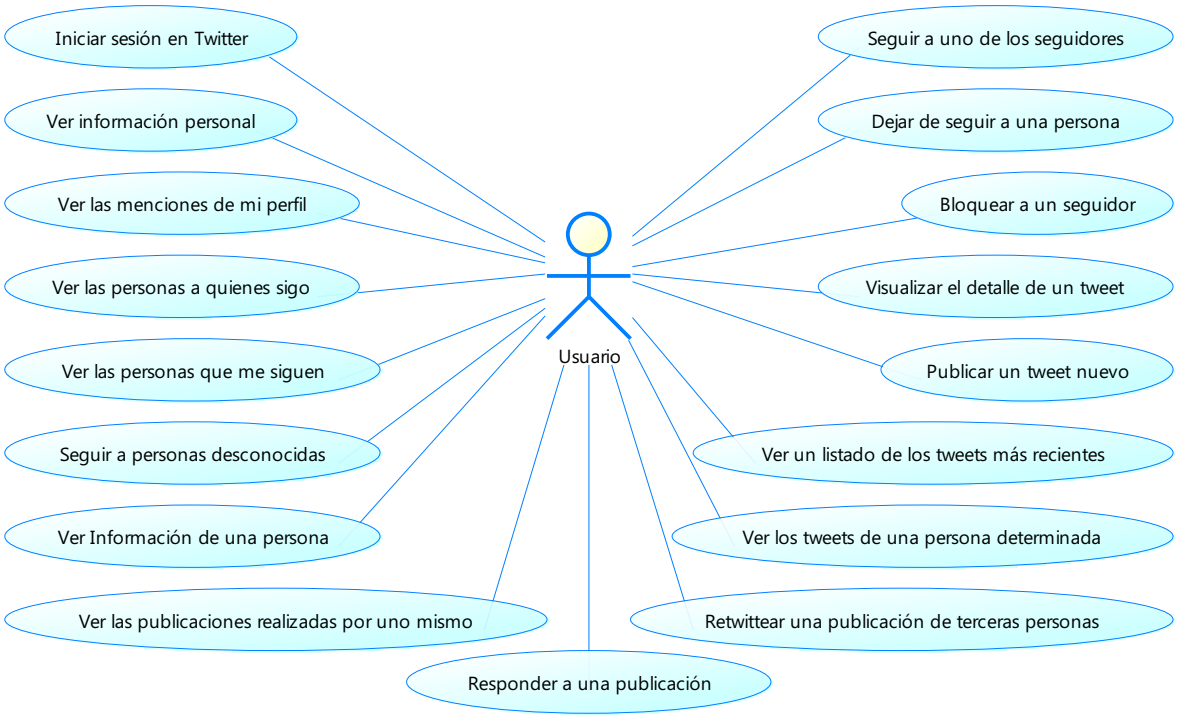



**Figura 3.2. Diagrama General de Casos de Uso de Tweet 4 ME**
Elaborado por: Carlos Andrés Oquendo

Esta definición general se encuentra plasmada en un artefacto denominado Modelo de Casos de Uso, que se incluirá como anexo de este documento.

Adicionalmente, para cada uno de los casos de uso presentados, se debe elaborar una descripción detallada. A continuación, y a manera de ejemplo, se presenta el detalle de uno de los casos de uso que aplican para el prototipo.

## User Interface Library for MIDP - UIL 4 MIDP
## Prototipo Práctico: Tweet 4 ME
## Caso de Uso: *Iniciar Sesión en Twitter*

### Descripción

Este caso de uso describe la forma en la cual los usuarios de Twitter pueden iniciar sesión utilizando la aplicación Tweet 4 ME.

### Actores

1. Usuario

### Precondiciones

- El Usuario dispone de una cuenta de usuario en Twitter
- La aplicación no se está ejecutando.

### Flujo de Eventos

1. El caso de uso comienza cuando el usuario desea ingresar a Twitter.
2. El usuario ejecuta la aplicación Tweet 4 ME.

3. La aplicación muestra la pantalla de inicio de sesión.
4. El usuario ingresa su nombre de usuario y contraseña. Cuando termina, selecciona la opción Aceptar.
5. La aplicación verifica que el usuario haya ingresado la información solicitada.
    - Si la información se encuentra completa.
        1. La aplicación se conecta a Twitter para validar la información.
            - Si la información es válida
                1. La aplicación muestra la pantalla de perfil personal.
            - Si la información no es válida
                1. La aplicación muestra un mensaje en el que indica que la información suministrada es incorrecta.
                2. El usuario acepta el mensaje.
    - Si la información no está completa
        1. La aplicación muestra un mensaje indicando al usuario que debe llenar toda la información solicitada.
        2. El usuario acepta el mensaje.
6. El caso de uso termina.

### Flujos Alternativos

N/A

### Post condiciones

- El usuario ha ingresado al servicio de Twitter con una sesión válida.

Todas las descripciones de los casos de uso correspondientes a la aplicación Tweet 4 ME, constan como anexos de este documento.

### 3.2.2. DISEÑO DE INTERFACES GRÁFICAS DE USUARIO DEL PROTOTIPO

Para soportar el cumplimiento de los casos de uso descritos en la sección 3.2.1., se ha diseñado un conjunto de vistas e interfaces de usuario para el prototipo. Este diseño está ilustrado en las Figuras 3.3, 3.4, 3.5, 3.6, 3.7, 3.8, 3.9 y 3.10, elaboradas por el autor.

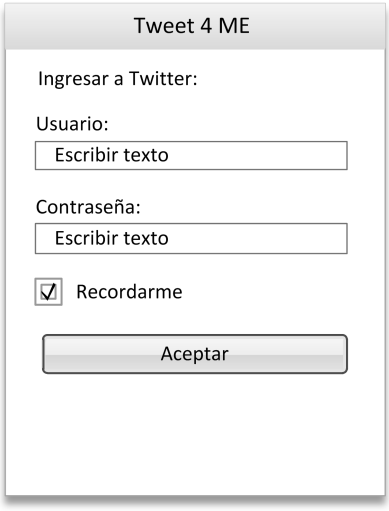



**Figura 3.3. Pantalla de Ingreso**


**Figura 3.4. Vista del Perfil del Usuario**

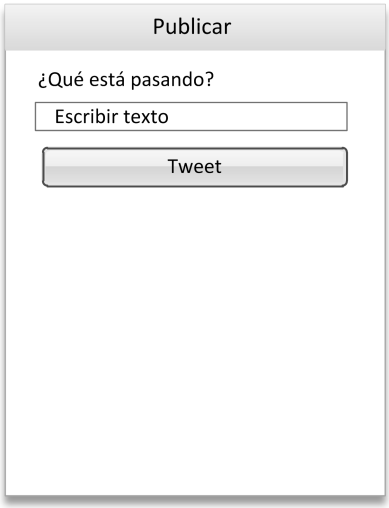



**Figura 3.5 Editor de Tweets**


**Figura 3.6 Listado de Personas**


**Figura 3.7 Detalle del Perfil de un Usuario**

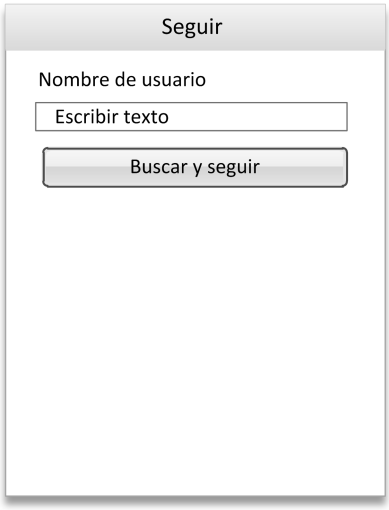



**Figura 3.8 Vista de Búsqueda de Usuarios**


**Figura 3.9 Listado de Tweets**


**Figura 3.10 Pantalla de Detalle de un Tweet**

### 3.2.3. CONSTRUCCIÓN DEL PROTOTIPO

Una vez que se encuentran definidas las funcionalidades y las vistas básicas que conformarán la solución, se puede ya empezar a construir el prototipo.

El desarrollo de las vistas de la aplicación se realizará utilizando la biblioteca de clases UIL4MIDP, materia de este documento, y por lo tanto se la incorporará como una referencia del proyecto de desarrollo.

Para soportar las funcionalidades relacionadas con el servicio Twitter se utilizará la biblioteca de clases TwAPIme, que es un componente de código abierto y de software libre, disponible para descarga en el sitio web http://www.twapime.com/.

Adicionalmente, para realizar algunas tareas sencillas de manejo de datos, se empleará el componente utilitario denominado HandyDB, que es una clase que simplifica de manejo del API de persistencia de Java Micro Edition. Este componente se encuentra en el dominio público y está disponible en http://sourceforge.net/projects/rms-db-class/.

Para estructurar y organizar de una manera lógica las clases y demás archivos y recursos que forman parte de la aplicación, se ha definido la estructura de paquetes que se enumera en la Tabla 3.3.

| No. | Paquete | Propósito |
|---|---|---|
| 1 | ec.edu.epn.fis.twitter.client | Contiene las clases principales de la aplicación. |
| 2 | ec.edu.epn.fis.twitter.client.login | Incorpora las clases requeridas para el proceso de inicio de sesión en el servicio. |
| 3 | ec.edu.epn.fis.twitter.client.resources | Contiene todos los recursos (imágenes e iconos) necesarios para complementar las vistas de la aplicación. |
| 4 | ec.edu.epn.fis.twitter.client.secret | Mantiene una clase donde se especifican los parámetros secretos de acceso al servicio Twitter. |
| 5 | ec.edu.epn.fis.twitter.client.user | Incluye todas las clases relacionadas con las vistas que forman parte de la aplicación. |
| 6 | ec.edu.epn.fis.twitter.client.util | Incluye clases que encapsulan procedimientos y funcionalidades útiles en todo el espectro de la aplicación. |

**Tabla 3.3. Estructura General de Paquetes del Prototipo**
Elaborado por: Carlos Andrés Oquendo

En la Figura 3.11 se encuentra una captura de pantalla donde se muestra la estructura del proyecto creado dentro del entorno de desarrollo.

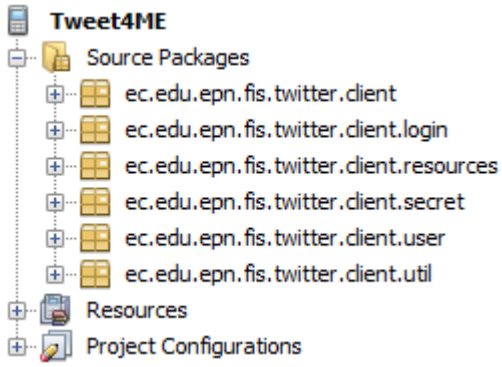



**Figura 3.11 Ejemplo de la Estructura de Paquetes del Prototipo**
Elaborado por: Carlos Andrés Oquendo

A continuación se incluye el código fuente elaborado para la vista de inicio de sesión, que corresponde a la clase `Login`. Las demás clases que conformen el prototipo Tweet 4 ME se escribirán de una manera similar.

```
package ec.edu.epn.fis.twitter.client.login;

import com.twitterapime.rest.Credential;
import com.twitterapime.rest.TweetER;
import com.twitterapime.rest.UserAccountManager;
import com.twitterapime.search.LimitExceededException;
import ec.edu.epn.fis.twitter.client.Environment;
import ec.edu.epn.fis.twitter.client.MainWindow;
import ec.edu.epn.fis.twitter.client.TweetForME;
import ec.edu.epn.fis.twitter.client.secret.APIValues;
import ec.edu.epn.fis.twitter.client.user.MainController;
import ec.edu.epn.fis.twitter.client.util.HandyDB;
import ec.edu.epn.fis.uil4midp.actions.ActionListener;
import ec.edu.epn.fis.uil4midp.components.controls.Button;
import ec.edu.epn.fis.uil4midp.components.controls.Label;
import ec.edu.epn.fis.uil4midp.components.controls.Switch;
import ec.edu.epn.fis.uil4midp.components.controls.TextBox;
import ec.edu.epn.fis.uil4midp.util.FontManager;
import ec.edu.epn.fis.uil4midp.views.Form;
import ec.edu.epn.fis.uil4midp.views.MessageDialog;
import ec.edu.epn.fis.uil4midp.views.ProgressDialog;
import java.io.IOException;
import javax.microedition.lcdui.TextField;

public class Login extends Form {

    private Label lblWelcome;
    private TextBox tbNombreUsuario;
    private TextBox tbContraseña;
    private Switch swRememberPassword;
    private Button btnLogin;

    private TweetForME midlet;
    private Environment env;

    public Login(final TweetForME midlet) {
        super("Tweet 4 ME");

        this.midlet = midlet;
        this.env = env = Environment.getInstance();
```

```
        setLeftButton("Salir", new ActionListener() {

            public void execute() {
                midlet.destroyApp(true);
            }
        });
    }

    public void initialize() {
        lblWelcome = new Label("Ingresar a Twitter");
        lblWelcome.setFont(FontManager.getBoldFont());
        addVisualComponent(lblWelcome);

        tbNombreUsuario = new TextBox("Usuario", 50, TextField.ANY | TextField.NON_PREDICTIVE);
        addVisualComponent(tbNombreUsuario);

        tbContraseña = new TextBox("Contraseña", 50, true);
        addVisualComponent(tbContraseña);

        swRememberPassword = new Switch("Recordar mis datos");
        addVisualComponent(swRememberPassword);

        btnLogin = new Button("Aceptar");
        btnLogin.setActionListener(new ActionListener() {

            public void execute() {
                // Verificar que se hayan ingresado nombre de usuario y contraseña.
                if (tbNombreUsuario.getText().length() == 0 ||
                                                    tbContraseña.getText().length() == 0) {
                    showDialog(new MessageDialog("Tweet 4 ME",
                                        "Debe ingresar su nombre de usuario y contraseña."));
                    return;
                }

                //Preparar el cuadro de progreso
                final ProgressDialog pdLogin = new ProgressDialog("Tweet 4 ME",
                                                                    "Iniciando sesión...");
                pdLogin.setDismissActionListener(new ActionListener() {
                    public void execute() {
                        try {
                            //Verificar si se debe almacenar la info
                            boolean shouldStoreData = swRememberPassword.isTurnedOn();
                            if (shouldStoreData) {
                                HandyDB.writeBoolean("T4M_PWSaved", true);
                                HandyDB.writeString("T4M_User", tbNombreUsuario.getText());
                                HandyDB.writeString("T4M_Pass", tbContraseña.getText());
                            } else {
                                HandyDB.delete("T4M_PWSaved");
                                HandyDB.delete("T4M_User");
                                HandyDB.delete("T4M_Pass");
                            }

                            //Añadir el nuevo controlador a la ventana.
                            controller.getWindow().setViewController(
                              new MainController((MainWindow) controller.getWindow(), midlet)
                            );
                        } catch (IOException ex) {
                            ex.printStackTrace();
                        }
                    }
                });

                //Mostrar el diálogo de progreso
                showDialog(pdLogin);
```

```
                //Iniciar sesión
                Thread t = new Thread(new Runnable() {

                    public void run() {
                        try {
                            Credential cr = new Credential(tbNombreUsuario.getText(),
                                    tbContraseña.getText(),
                                    APIValues.APP_CONSUMER_KEY,
                                    APIValues.APP_CONSUMER_SECRET);

                            UserAccountManager manager = UserAccountManager.getInstance(cr);
                            env.setUserAccountManager(manager);

                            if (manager.verifyCredential()) {
                                env.setTwitter(TweetER.getInstance(manager));
                                pdLogin.close();
                            }  else {
                                pdLogin.close(false);
                                showDialog(new MessageDialog("Tweet 4 ME",
                     "Imposible iniciar sesión. Revise su nombre de usuario y contraseña."));
                            }
                        } catch (IOException ex) {
                            ex.printStackTrace();
                            pdLogin.close(false);
                            showDialog(new MessageDialog("Tweet 4 ME",
                                "Conexión a Internet falló. Imposible conectar a Twitter."));
                        } catch (LimitExceededException ex) {
                            ex.printStackTrace();
                            pdLogin.close(false);
                            showDialog(new MessageDialog("Tweet 4 ME",
                                            "Límites de conexión con Twitter excedidos."));
                        } catch (Exception ex) {
                            ex.printStackTrace();
                            pdLogin.close(false);
                            showDialog(new MessageDialog("Tweet 4 ME",
                     "Imposible iniciar sesión. Revise su nombre de usuario y contraseña."));
                        }
                    }
                }, "T4M_LoginThread");
                t.start();
            }
        });
        addVisualComponent(btnLogin);

        String userName;
        String passWord;
        boolean isPasswordSaved = HandyDB.readBoolean("T4M_PWSaved");
        if (isPasswordSaved) {
            userName = HandyDB.readString("T4M_User");
            passWord = HandyDB.readString("T4M_Pass");

            tbNombreUsuario.setText(userName);
            tbContraseña.setText(passWord);
            swRememberPassword.setTurnedOn(true);
        }
    }
}
```

Concluida la implementación de todas las clases que conforman el prototipo, se deberá crear una versión compilada de la aplicación Tweet 4 ME, que es la que se utilizará para ejecutar las pruebas de la solución UIL4MIDP.

Esta versión del prototipo Tweet 4 ME incluye los siguientes entregables:

- Archivos Compilados:
  - Tweet4ME.jad – Descriptor del MIDlet.
  - Tweet4ME.jar – Archivo ejecutable de la aplicación.
- Código Fuente

  Incluye todo el código fuente del prototipo, los archivos de recursos y el proyecto configurado para el entorno de desarrollo NetBeans IDE.
- Librerías
  - TwAPIme.jar – Biblioteca de clases para acceso al servicio Twitter.
  - UIL4MIDP.jar – Biblioteca de clases para manejo de la interfaz gráfica.

En la Figura 3.12 se muestra la pantalla de inicio de sesión de la aplicación ejecutándose sobre el emulador parte del SDK de Java Micro Edition.


**Figura 3.12 Pantalla de Inicio de Sesión de Tweet 4 ME en Emulador**
Elaborado por: Carlos Andrés Oquendo

## 3.3. PRUEBAS DEL PROTOTIPO

Las pruebas que se realicen al prototipo Tweet 4 ME permitirán determinar si las premisas definidas para la solución UIL4MIDP son válidas, y de esta manera, si se cumple con los objetivos definidos para esta solución.

### 3.3.1. ESPECIFICACIÓN DE PRUEBAS

Para evidenciar el correcto funcionamiento de la solución UIL4MIDP, se realizarán las siguientes comprobaciones generales:

1. Correcto ajuste de las dimensiones de las vistas en función del tamaño de la pantalla del dispositivo.
2. Ajuste de la posición de los controles.
3. Adaptación de los controles al tipo de letra disponible en el entorno Java ME del dispositivo.

### 3.3.2. CRITERIOS DE COMPARACIÓN

Los criterios de comparación definen un estado base sobre el cual se puede realizar las pruebas y relaciones que hayan sido definidas.

Los criterios que se aplicarán, se tomarán a partir de las características del emulador del SDK de Java Micro Edition proporcionado por Sun Microsystems y de la apariencia que la aplicación genera en este entorno.

En la Tabla 3.4 se definen las características correspondientes al emulador presente en el SDK.

| Dispositivo | Sistema Operativo | Versión de MIDP | Versión de CLDC | Tamaño de Pantalla |
|---|---|---|---|---|
| **DefaultColorPhone, DefaultCldcPhone** | Java ME Runtime | 2.0 | 1.0 | 240 x 320 px. |

**Tabla 3.4. Características Generales del Entorno de Control**
Elaborado por: Carlos Andrés Oquendo

Para ejecutar las comprobaciones definidas, principalmente de aspectos visuales, se incluyen, en las Figuras 3.13, 3.14, 3.15 y 3.16, capturas de pantalla de algunas de las vistas de la aplicación ejecutándose sobre el emulador. Estas imágenes son elaboradas por el autor.


Figura 3.13. Vista de Formulario de Ingreso


Figura 3.14. Vista de Formulario

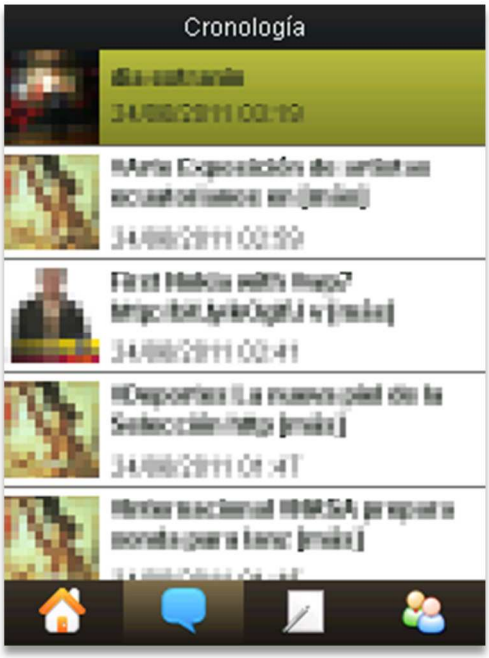



Figura 3.15. Vista de Lista

### 3.3.3. EJECUCIÓN EN LOS DISPOSITIVOS DE PRUEBA

La ejecución en los dispositivos de prueba se realiza tomando en cuenta las características de cada uno de los dispositivos enumerados en la sección 3.1.3., en especial, el tamaño de la pantalla.

Todas las imágenes que aparecen en las secciones 3.3.3.1, 3.3.3.2 y 3.3.3.3 han sido elaboradas por el autor.

### 3.3.3.1. VISTA DE FORMULARIO DE INGRESO


Figura 3.16. Formulario de Ingreso en Nokia 5530 XpressMusic


Figura 3.17. Formulario de Ingreso en Sony Ericsson W302


Figura 3.18. Formulario de Ingreso en Samsung GT E2121L


Figura 3.19. Formulario de Ingreso en Nokia N8


Figura 3.20. Formulario de Ingreso en Samsung GT M2310


Figura 3.21. Formulario de Ingreso en Sony Ericsson W200

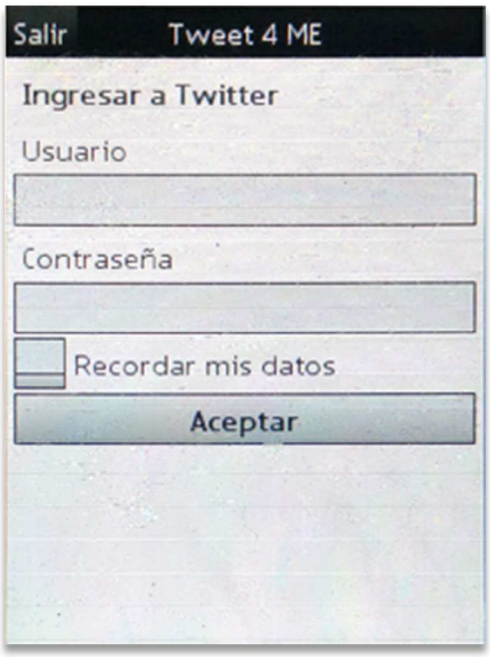



**Figura 3.22. Formulario de Ingreso en Nokia 6300**

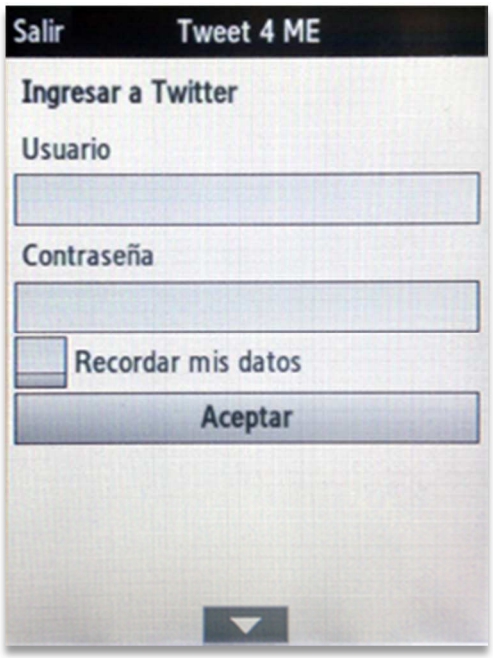



**Figura 3.23. Formulario de Ingreso en Samsung GT B3410**


**Figura 3.24. Formulario de Ingreso en Motorola U9**

### 3.3.3.2. VISTA DE FORMULARIO GENÉRICO

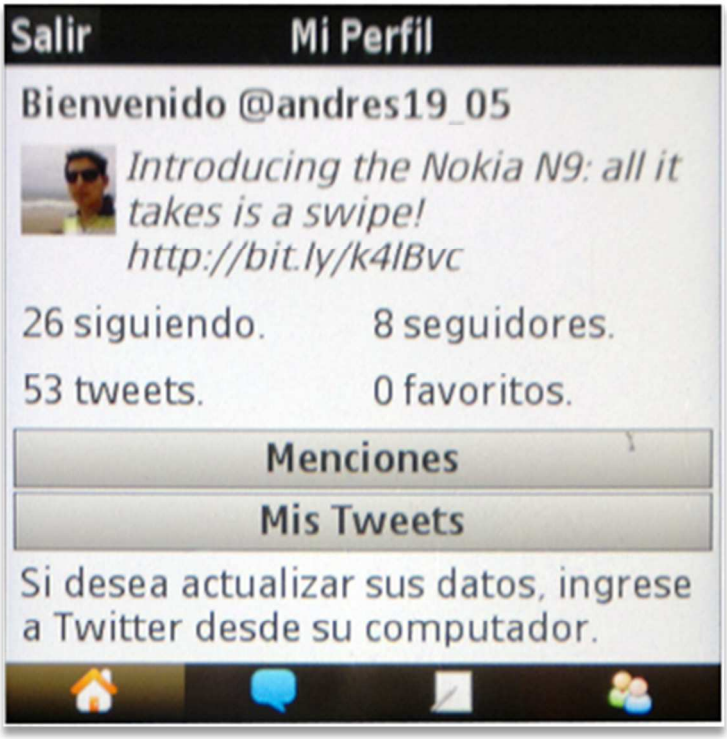



**Figura 3.25. Formulario Genérico en Nokia 5530 XpressMusic**


**Figura 3.28. Formulario Genérico en Nokia N8**

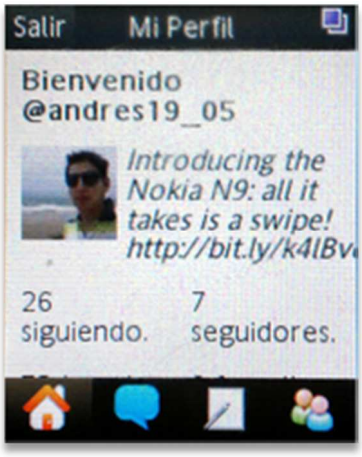



**Figura 3.26. Formulario Genérico en Sony Ericsson W302**


**Figura 3.27. Formulario Genérico en Samsung GT E2121L**

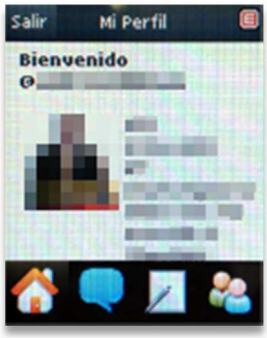



**Figura 3.29. Formulario Genérico en Samsung GT M2310**

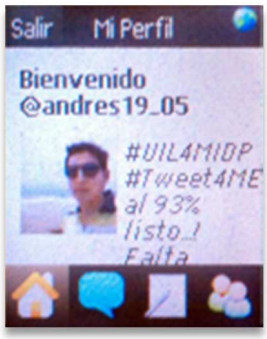



**Figura 3.30. Formulario Genérico en Sony Ericsson W200**


Figura 3.31. Formulario Genérico en Nokia 6300


Figura 3.32. Formulario Genérico en Samsung GT B3410


Figura 3.33. Formulario Genérico en Motorola U9

### 3.3.3.3. VISTA DE LISTA

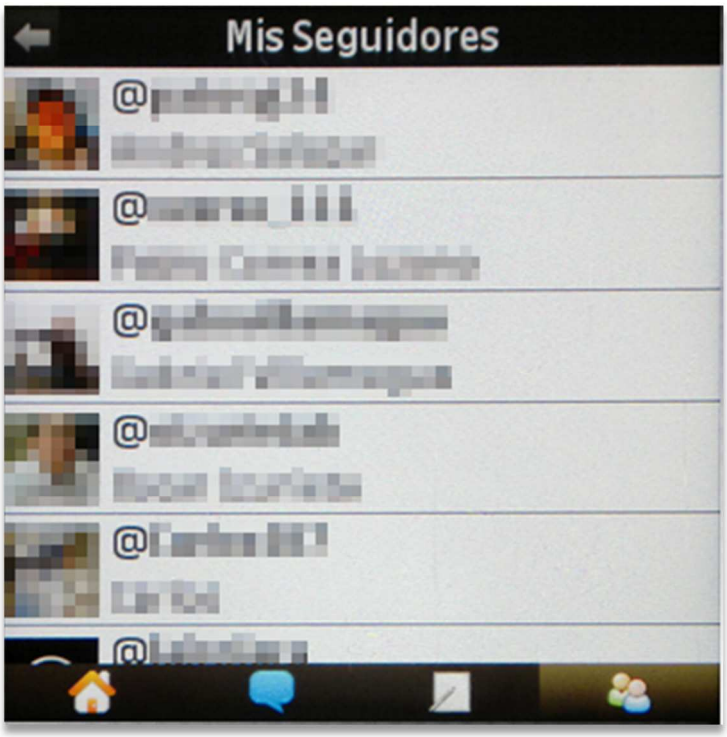



Figura 3.34. Lista en Nokia 5530 XpressMusic

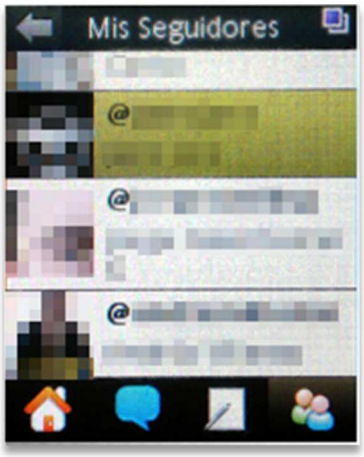



Figura 3.35. Lista en Sony Ericsson W302

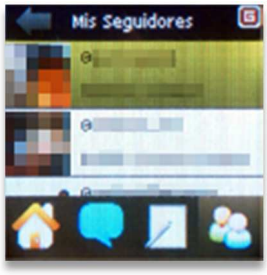



Figura 3.36. Lista en Samsung GT E2121L

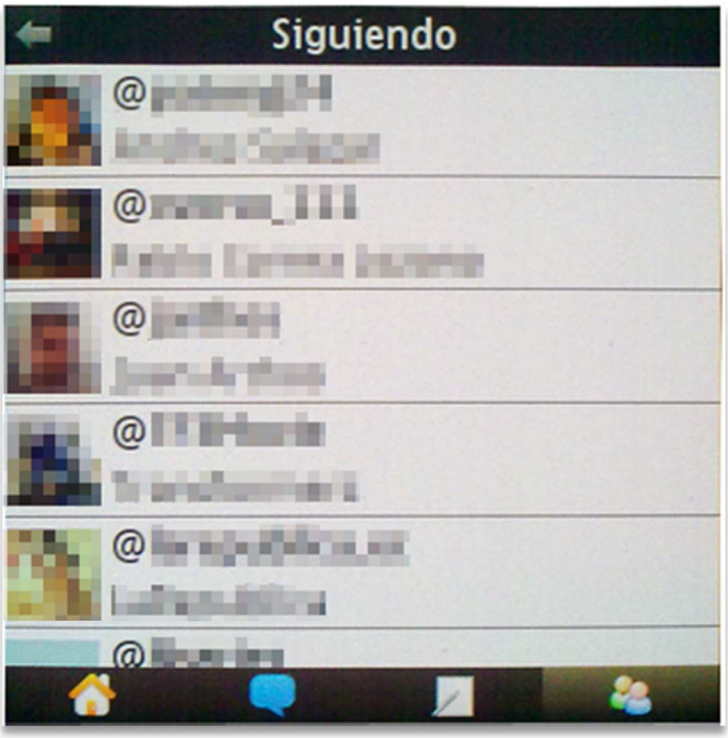



Figura 3.37. Lista en Nokia N8

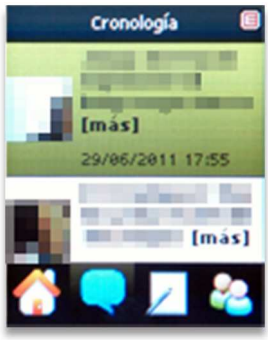



Figura 3.38. Lista en Samsung GT M2310

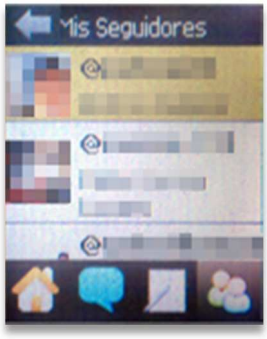



Figura 3.39. Lista en Sony Ericsson W200

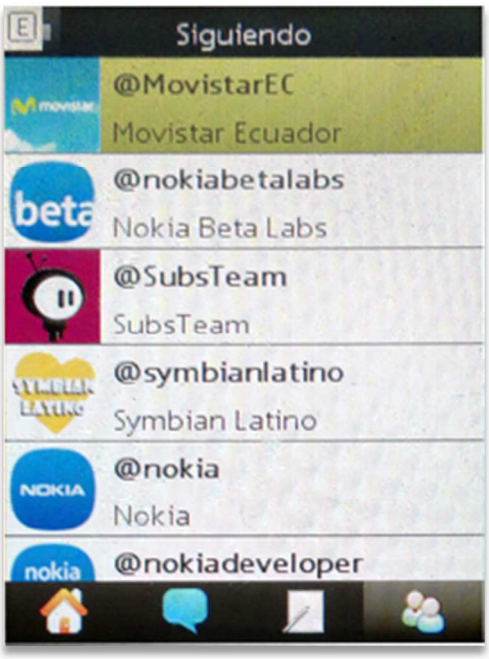



Figura 3.40. Lista en Nokia 6300

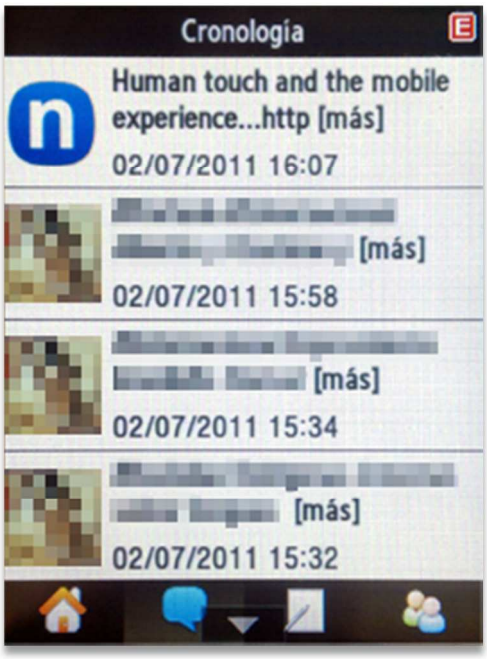



Figura 3.41. Lista en Samsung GT B3410

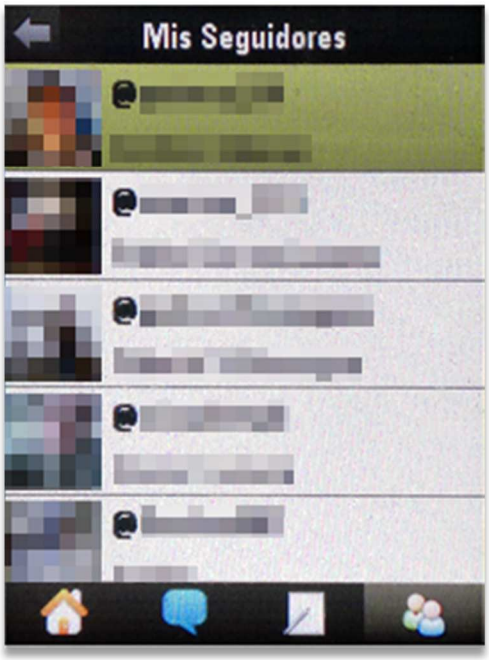



Figura 3.42. Lista en Motorola U9

## 3.4. ANÁLISIS DE RESULTADOS

De las imágenes presentadas en la sección 3.3.3 y considerando las imágenes que forman parte de los criterios de comprobación, definidas en la sección 3.3.2 de este documento, es posible evidenciar lo siguiente:

- Las interfaces gráficas de usuario incorporadas como parte del prototipo mantienen una apariencia uniforme en los dispositivos donde la aplicación se ejecuta.

  En contraste, en las Figuras 3.43 y 3.44 se puede observar la apariencia disímil con la que una pantalla programada utilizando el API estándar LCDUI se presenta, cuando la aplicación se ejecuta en dos ambientes con características diferentes.

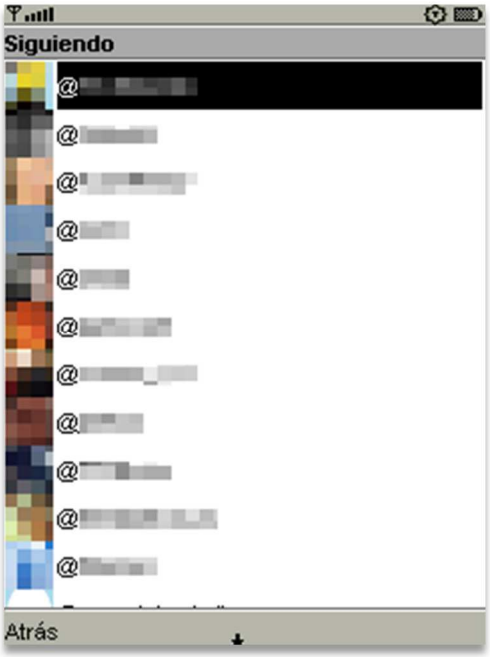



**Figura 3.43. Lista LCDUI en Emulador del SDK Estándar**
Elaborado por: Carlos Andrés Oquendo

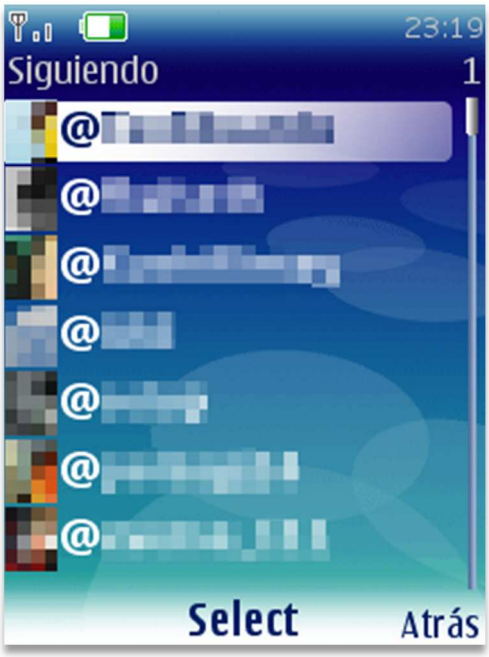



**Figura 3.44. Lista LCDUI en Emulador del SDK Nokia Series 40**
Elaborado por: Carlos Andrés Oquendo

- Los controles de usuario de las vistas se ajustan correctamente al ancho de la pantalla del dispositivo donde se ejecuta la aplicación.

Esta situación se aprecia con mayor claridad en la barra de *pestañas* presente en algunas de las vistas, y que se puede visualizar en la Figura comparativa 3.45.

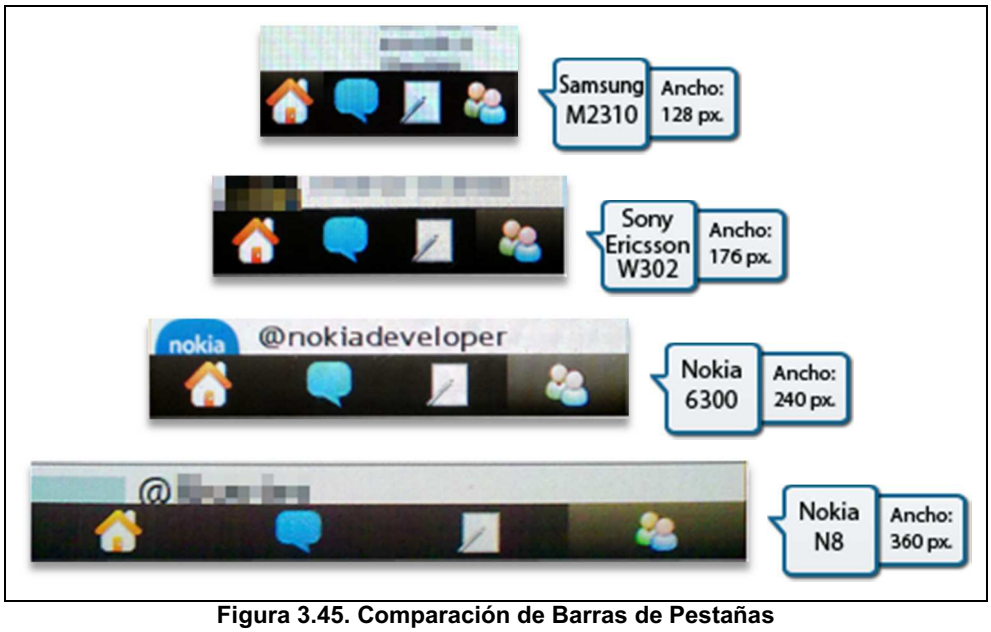



**Figura 3.45. Comparación de Barras de Pestañas**
Elaborado por: Carlos Andrés Oquendo

- El alto de los controles se ajusta automáticamente en función del tipo de letra disponible en cada uno de los dispositivos. Esta situación se puede apreciar al comparar el botón presente en la vista de formulario de ingreso, y que se puede visualizar en la Figura comparativa 3.46.


**Figura 3.46. Comparación de Botones de Formulario de Ingreso**
Elaborado por: Carlos Andrés Oquendo

- El texto presente en los controles se dibuja utilizando el tipo de letra estándar del sistema operativo del dispositivo. En la Figura 3.47 se puede apreciar con claridad esta situación.

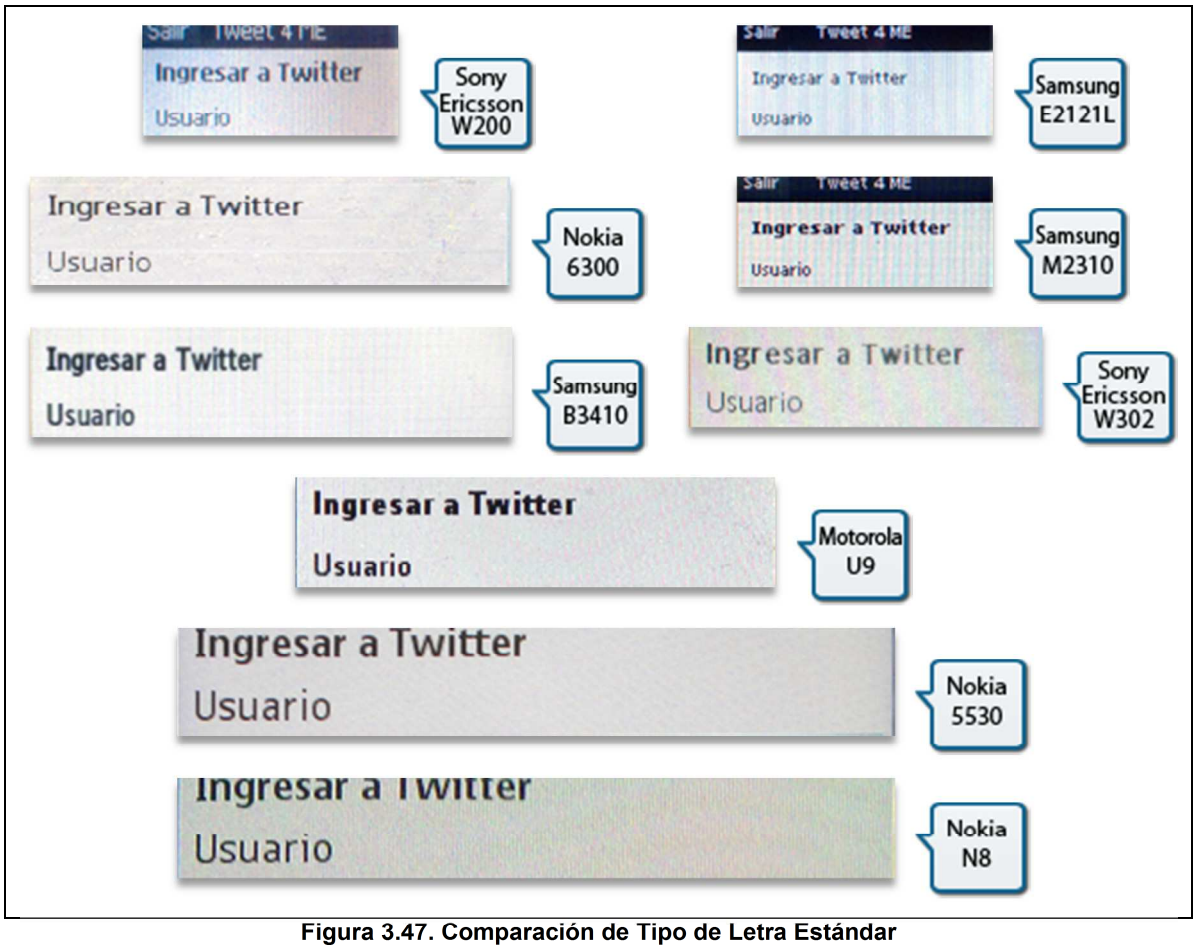



**Figura 3.47. Comparación de Tipo de Letra Estándar**
Elaborado por: Carlos Andrés Oquendo

- El texto que se encuentra dentro de los controles se ajusta y se divide en líneas de manera automática tomando en cuenta el espacio disponible en la pantalla para ubicarlo. En la Figura 3.48 se puede visualizar este comportamiento de la solución.

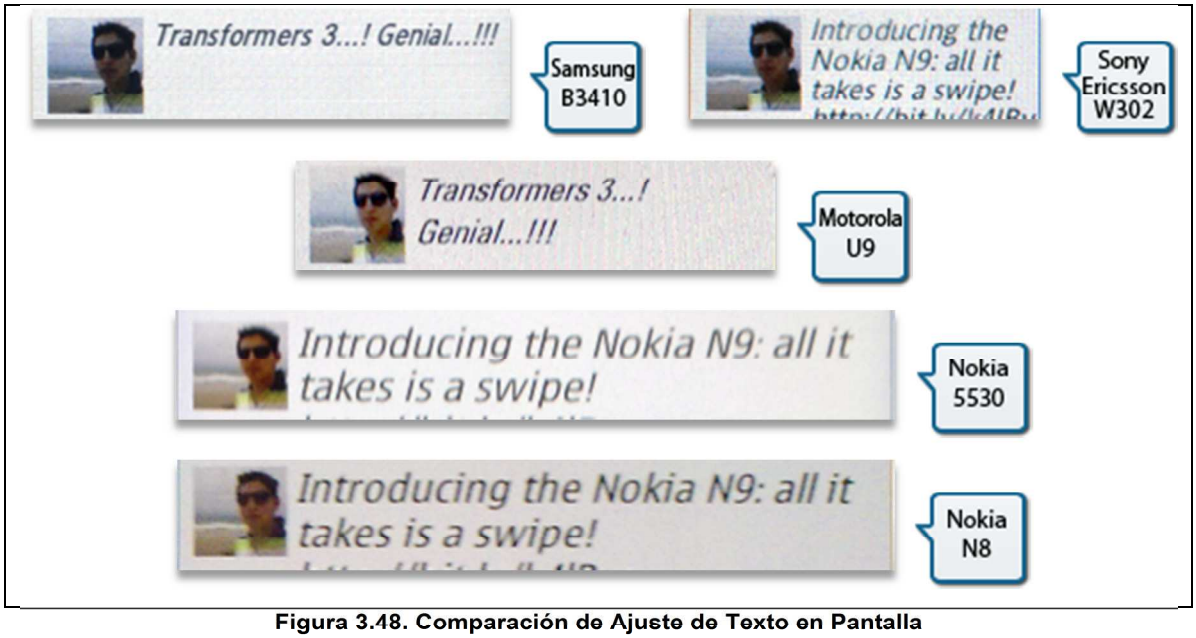



**Figura 3.48. Comparación de Ajuste de Texto en Pantalla**
Elaborado por: Carlos Andrés Oquendo

- En ocasiones, el sistema operativo de los dispositivos podría mostrar indicadores de actividad o elementos adicionales. Estos objetos no pueden ser controlados por UIL4MIDP y podrían aparecer en lugares arbitrarios de la pantalla. En la Figura comparativa 3.49 se aprecia este comportamiento.

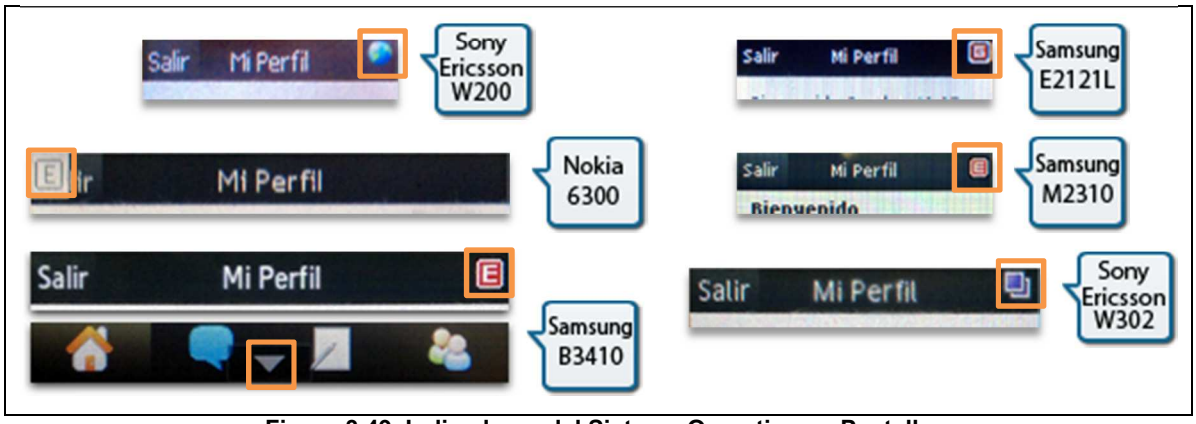



**Figura 3.49. Indicadores del Sistema Operativo en Pantalla**
Elaborado por: Carlos Andrés Oquendo

- Finalmente, tanto en las capturas de pantalla que conforman los criterios de comprobación (incluidas en la sección 3.3.2) así como en las capturas de pantalla presentadas en la sección 3.3.3, se puede evidenciar que la posición en la que los controles de usuario se ubican se adapta en cada uno de los dispositivos.

# CAPÍTULO 4. CONCLUSIONES Y RECOMENDACIONES

Como parte final del proceso de desarrollo de la solución User Interface Library for MIDP – UIL4MIDP, y de los demás elementos accesorios, como el prototipo demostrativo Tweet 4 ME, es necesario establecer un conjunto de conclusiones y recomendaciones que recojan las experiencias aprendidas.

## 4.1. CONCLUSIONES

- Java Micro Edition es una plataforma operativa práctica sobre la cual se pueden desarrollar aplicaciones móviles orientadas a un segmento de mercado compuesto por una gran cantidad de dispositivos de varias marcas y modelos, especialmente de gama media y baja. Sin embargo, es necesario destacar que el uso de esta plataforma impone una serie de restricciones respecto a las funcionalidades con las que se puede elaborar una aplicación.

- A través del desarrollo de esta solución, se ha logrado entender y aplicar de mejor manera la metodología de desarrollo de software Open UP, que ha permitido llevar a cabo las tareas requeridas de una manera organizada, definida y con un nivel de detalle adecuado a la dimensión del proyecto.

- Se ha demostrado, a través de la visualización de la ejecución del prototipo Tweet 4 ME, que la solución UIL4MIDP permite construir aplicaciones cuyas interfaces gráficas de usuario funcionan de manera estable y uniforme en dispositivos de características y prestaciones variadas.

- La solución UIL4MIDP aporta positivamente en la reducción del tiempo en el que se diseñan e implementan las interfaces gráficas de usuario de una aplicación. Se estima que la reducción general de tiempo alcanza por lo menos un 15% al considerar que la solución proporciona componentes estandarizados que, aun cuando se usan con la configuración predeterminada, permiten elaborar vistas que poseen una gran riqueza visual, y que están listas para funcionar en una amplia gama de dispositivos.

- Para un desarrollador, elaborar una interfaz gráfica con una funcionalidad similar a la expuesta por la solución, con los mecanismos estándar de la

plataforma Java ME, puede ser un trabajo demandante y repetitivo, que a la larga afecta a la productividad.

- El proceso de desarrollo de una biblioteca de clases tiene mayor complejidad que el de una aplicación diseñada para usuarios finales debido a la naturaleza abstracta y genérica que es intrínseca a este tipo de productos de software.

- Cuando se desarrolla una biblioteca de clases puede ser muy complicado utilizar técnicas de modelado y diseño basadas en casos de uso, debido a que las acciones y procesos que se deben componer se encuentran a un nivel muy abstracto.

- La estimación de tiempos y costos de un proyecto debe realizarse minuciosamente tratando de descubrir todos los elementos que intervienen en su desenvolvimiento, o que podrían afectarlo, para lograr que sean lo más apegados a la realidad.

- Es necesario mencionar que, en el caso de este proyecto, se intentó cumplir con el tiempo establecido a la ejecución de las tareas. Sin embargo, esto no siempre fue posible debido a imprevistos que alteraron de una u otra manera el proceso de desarrollo.

## 4.2. RECOMENDACIONES

- La solución UIL4MIDP puede seguir evolucionando más allá del alcance definido para el proyecto de titulación. De esta forma, los componentes y funcionalidades mejorarían y permitirían tener un producto más robusto y útil. Las siguientes son funcionalidades que podrían incorporarse en futuras versiones de la solución:

    - Inclusión de soporte para dispositivos con pantalla táctil.
    - Incorporación de controles de usuario adicionales o especializaciones de los existentes.
    - Soporte nativo a métodos de ingreso de texto.
    - Adición de niveles adicionales de extensibilidad que permitan la ejecución de más eventos personalizados.

- Para el desarrollo de bibliotecas de clases en general, puede resultar ventajoso empezar a construir, de manera simultánea, una aplicación que consuma las funcionalidades incorporadas en la biblioteca. De esta manera, se puede evidenciar con prontitud si es correcto el diseño de la biblioteca de clases y si las funcionalidades están bien implementadas y enfocadas. Si es necesario hacer algún cambio en la biblioteca de clases, el impacto asociado es evidentemente mucho menor que el que se produciría si la biblioteca de clases tiene un nivel de completitud más avanzado.

- Como buena práctica para el proceso de desarrollo de cualquier tipo de aplicación o solución, se debería contar con un repositorio (local o remoto) donde el código fuente pueda ser centralizado, integrado y resguardado. De esta manera, se puede lograr la cooperación de los miembros del equipo de desarrollo así como aplicar esquemas de versionamiento y control de cambios.

- De manera similar, es recomendable que los documentos y artefactos elaborados se encuentren versionados y respaldados en repositorios adecuados, locales o remotos, que permitan salvaguardar su integridad y disponibilidad.

- Para la construcción de aplicaciones, y con el fin de promover la reusabilidad de software, se debería siempre analizar con buen criterio la posibilidad de utilizar librerías de software de terceros, especialmente de la solución UIL4MIDP, desarrollada como parte de este proyecto.

- El uso de librerías debería evaluarse tomando en cuenta su calidad general, las posibilidades de aprovechamiento de las funcionalidades presentadas, y de pruebas de funcionamiento que certifiquen su verdadera utilidad para solventar los requerimientos definidos para una aplicación.

- Al realizar las estimaciones de tiempos podría resultar conveniente incorporar un porcentaje adicional de tiempo en cada tarea que, en caso de requerirse, permita solventar situaciones imprevistas, evitando que el calendario o los costos del proyecto se alteren sustancialmente.

# BIBLIOGRAFÍA

AmericaEconomia. (10 de Febrero de 2011). *Ecuador: penetración de telefonia movil llega al 105%*. Recuperado el Julio de 2011, de AETecno: http://tecno.americaeconomia.com/noticias/ecuador-penetracion-de-telefonia-movil-llega-al-105

B'Far, R. (2005). *Mobile Computing Principles.* Cambridge: Cambridge University Press.

Blanco, E. (08 de Marzo de 2011). *OSS Watch - Dual Licensing*. (University of Oxford) Recuperado el 21 de Mayo de 2011, de http://www.oss-watch.ac.uk/resources/duallicence2.xml

Eclipse Foundation. (2010). *Open UP*. Recuperado el 03 de Julio de 2010, de http://epf.eclipse.org/wikis/openup/

Enough Software. (2010). *J2ME Polish*. (Enough Software) Recuperado el 17 de Marzo de 2010, de Introduction: http://www.j2mepolish.org/cms/leftsection/introduction.html

Knudsen, J., & Li, S. (2005). *Beginning J2ME: From Novice to Professional, Third Edition.* Apress.

Mahmoud, Q. (2001). *Learning Wireless Java.* O'Reilly.

mobiThinking. (Julio de 2011). *Global Mobile Statistics 2011*. Recuperado el Julio de 2011, de mobiThinking: http://www.mobithinking.com/mobile-marketing-tools/latest-mobile-stats

Motorola Mobility, Inc. (2011). *MOTODEV: Platforms & Application Environments: Java*. Recuperado el 05 de Julio de 2011, de MOTODEV, The Motorola Developer Network: http://developer.motorola.com/platforms/java/

Mourao, E. (2011). *Twitter API ME Wiki*. Recuperado el Marzo de 2011, de TwAPIme: http://www.twapime.com/

Nokia. (2011). *Device Specifications*. Recuperado el Junio de 2011, de Nokia Developer: http://www.developer.nokia.com/Devices/Device_specifications/

Nokia. (2011). *Wiki*. Recuperado el 2011, de Nokia Developer: http://www.developer.nokia.com/Community/Wiki/Wiki_Home

Oracle Corporation. (2010). *Java ME Technology - LWUIT*. (Oracle Corporation) Recuperado el 17 de Marzo de 2010, de Oracle, Sun Developer Network (SDN): http://java.sun.com/javame/technology/lwuit/

Oracle Corporation. (2011). *Java For Mobile Devices*. Recuperado el 16 de Marzo de 2011, de Oracle Wireless Java Client:

http://www.oracle.com/technetwork/java/javame/javamobile/overview/getstarted/index.html

Oracle Corporation. (2011). *Java ME Technology Overview*. Recuperado el 25 de Marzo de 2011, de http://www.oracle.com/technetwork/java/javame/java-me-overview-402920.html

Samsung. (2011). *Teléfonos Celulares*. Recuperado el 29 de Junio de 2011, de Samsung Latin: http://www.samsung.com/latin/consumer/type/productselector.do?group=mobile-phones&type=mobile-phones

Sony Ericsson Mobile Communications AB. (2011). *Phone Gallery*. Recuperado el 18 de Junio de 2011, de Developer World: http://developer.sonyericsson.com/wportal/devworld/phones/phone-gallery

The Apache Software Foundation. (Enero de 2004). *Apache License, Version 2.0*. (The Apache Software Foundation) Recuperado el 21 de Mayo de 2011, de http://www.apache.org/licenses/LICENSE-2.0.html

Twitter. (2011). *Documentation*. Recuperado el Marzo de 2011, de Twitter Developers: https://dev.twitter.com

Varios. (2008). *J4ME*. Recuperado el 17 de Marzo de 2010, de http://code.google.com/p/j4me/

Wells, M. J. (2005). *J2ME Game Programming.* Thomson Course Technology.

Wikimedia Foundation. (26 de Mayo de 2010). *Connected Device Configuration*. Recuperado el 03 de Julio de 2010, de Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/Connected_Device_Configuration

Wikimedia Foundation. (27 de Junio de 2010). *Connected Limited Device Configuration*. Recuperado el 03 de Julio de 2010, de Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/Connected_Limited_Device_Configuration

Wikimedia Foundation. (29 de Febrero de 2010). *Java Platform, Micro Edition.* Recuperado el 02 de Febrero de 2010, de Wikipedia, The Free Encyclopedia: http://en.wikipedia.org/wiki/Java_Platform,_Micro_Edition

# GLOSARIO

- **API**: Acrónimo de *Application Programming Interface* (Interfaz de Programación de Aplicaciones). Conjunto de procedimientos, funciones y métodos que pueden ser utilizados por otros componentes y programas para ejecutar alguna actividad.
- **CDC**: Acrónimo de *Connected Device Configuration* (Configuración de Dispositivo Conectado).
- **CLDC**: Acrónimo de *Connected, Limited Device Configuration* (Configuración de Dispositivo Conectado y Limitado).
- **Gradiente de Color**: Rango de colores dependientes de su posición que se utilizan para rellenar una región en la pantalla.
- **Javadoc**: Tecnología para la generación de la documentación perteneciente a un API. El producto se genera en formato HTML a partir del código fuente Java.
- **JCP**: Acrónimo de *Java Community Process* (Proceso Comunitario Java). Proceso estandarizado y formal que permite a partes interesadas participar en la definición de características y funcionalidades de la plataforma Java.
- **JME**: Acrónimo de *Java Micro Edition*. Similar a J2ME.
- **JSR**: Acrónimo de *Java Specification Request* (Solicitud de Especificaciones Java). Documento formal que describe las propuestas de adición de tecnologías a la plataforma Java.
- **JVM**: Acrónimo de *Java Virtual Machine* (Máquina Virtual de Java). Máquina virtual capaz de ejecutar el código fuente compilado (bytecode) de programas Java.
- **Keypad**: Teclado de un dispositivo móvil. Puede ser numérico (según el estándar ITU) o alfanumérico, dependiendo del factor de forma del dispositivo.
- **KVM**: Acrónimo de *Kilobyte Virtual Machine* (Máquina Virtual Kilobyte). Máquina virtual Java diseñada por Sun Microsystems, derivada de la JVM y optimizada para su uso en equipos pequeños y con poca memoria (en el orden de los Kilobytes).

- **LCDUI**: Acrónimo informal de *Limited Capability Device User Interface* (Interfaz de Usuario para Dispositivos de Capacidad Limitada). Representa a la tecnología estándar de diseño de interfaces gráficas de usuario para el perfil MIDP.
- **Microblogging**: Publicación electrónica de información de tamaño reducido en forma de frases cortas, imágenes individuales, enlaces de hipertexto, etc.
- **MIDlet**: Aplicación que utiliza el perfil MIDP y la configuración CLDC de la plataforma Java Micro Edition para funcionar.
- **MIDP**: Acrónimo de *Mobile Information Device Profile* (Perfil de Dispositivo de Información Móvil).
- **Tema Visual**: Definición en la que se incluyen detalles de apariencia gráfica que permiten personalizar la forma en que una aplicación se muestra.
- **Twitter**: Servicio de *microblogging* y red social que permite a sus usuarios enviar y recibir mensajes cortos de hasta 140 caracteres.

# ANEXOS

Los anexos que se enumeran a continuación, se encuentran en el disco compacto que acompaña a este documento.

**Anexo 1.** Documento de Visión de User Interface Library for MIDP.

**Anexo 2.** Glosario de User Interface Library for MIDP.

**Anexo 3.** Plan del Proyecto User Interface Library for MIDP.

**Anexo 4.** Lista de Elementos de Trabajo de User Interface Library for MIDP.

**Anexo 5.** Lista de Riesgos de User Interface Library for MIDP.

**Anexo 6.** Requerimientos Globales del Sistema de UIL4MIDP.

**Anexo 7.** Modelo de Casos de Uso de User Interface Library for MIDP

**Anexo 8.** Especificaciones de Casos de Uso de UIL4MIDP.

**Anexo 9.** Cuaderno de Arquitectura

**Anexo 10.** Licencia Apache 2.0

**Anexo 11.** Contrato de Licencia de Usuario Final (CLUF) de UIL4MIDP

**Anexo 12.** Diagramas de Diseño de UIL4MIDP (Formato Power Designer)

**Anexo 13.** Presupuesto del Proyecto User Interface Library for MIDP

**Anexo 14.** Modelo y Descripciones de Casos de Uso del Prototipo Tweet 4 ME.

**Anexo 15.** Diseño de Interfaces de Tweet 4 ME.