Mobile Information Device Profile (MIDP)

Florina Almenárez Mendoza
Celeste Campo
Departamento de Ingeniería Telemática
Universidad Carlos III de Madrid
florina@it.uc3m.es, celeste@it.uc3m.es

Parte de este material se basa en transparencias de Natividad Martínez Madrid (nati@it.uc3m.es)

Objetivo

- Aprender a desarrollar aplicaciones utilizando el API de MIDP 2.0

Bibliografía:

- ***J2ME : Java 2 micro edition : manual de usuario y tutorial***. Froufe, Agustín y Jorge, Patricia. Ra-Ma. [2004]. L/S 004.438 JAVA FRO, L/D 004.438 JAVA FRO. Capítulos 6, 9 al 12.
- ***Especificación de MIDP 2.0*** (JSR 118). Disponible en `http://www.jcp.org`
- ***Wireless Java Programming with Java 2 Micro Edition***. Feng, Yu and Zhu, Jun. SAMS [2001]. L/D 004.438 JAVA FEN. Capítulos 5 al 9.

Índice

- **Generalidades y conceptos básicos**
  - MIDlets y MIDlet Suite
    - desarrollo y despliegue
- Librerías de MIDP
- Interfaz de usuario
  - API de alto nivel
  - API de bajo nivel
  - API de juegos
- Conectividad
- Almacenamiento persistente

Versiones de MIDP

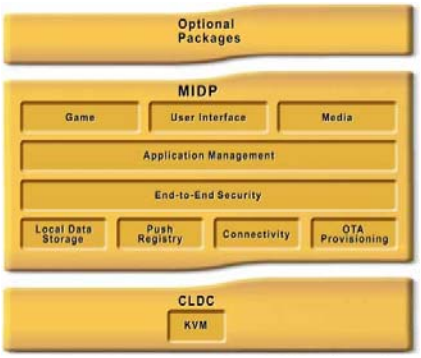
- **MIDP 1.0**
  - JSR 30
  - Final Release: Sep, 2000
- **MIDP 2.0**
  - JSR 118
  - Final Release: Nov, 2002
  - Final Release 2: Jun, 2006
- **MIDP 3.0**
  - JSR 271
  - "Final Release": Dic, 2009
  - MIDlets en CLDC, CDC, y OSGi

Generalidades

- **Requisitos hardware:**
  - Memoria:
    - 256 KB de memoria no volátil para los componentes MIDP
    - 8 KB de memoria no volátil para creación de datos persistentes
    - 128 KB de memoria volátil para la ejecución de Java
  - Pantalla:
    - Tamaño: 96x54
    - Profundidad: 1 bit
    - Aspecto pixel 1:1
  - Entrada, uno o más de los siguientes mecanismos:
    - Teclado “one-handed” o “two-handed”
    - Pantalla táctil
  - Conectividad:
    - Limitada, típicamente *wireless*
  - Sonido:
    - Tonos, vía hardware dedicado o algoritmo software

Alcance

- Define el conjunto de APIs disponibles para el desarrollo de aplicaciones portables entre dispositivos móviles.
- MIDP cubre:
  - ciclo de vida de la aplicación
  - interfaz de usuario
  - soporte de red
  - almacenamiento persistente
  - sonidos
  - juegos en 2D
  - seguridad extremo a extremo
  - timers, excepciones, …



Alcance (II)

- MIDP no cubre:
  - descarga y gestión de aplicaciones (MIDlets) en los dispositivos.
  - seguridad a bajo nivel.
  - gestión de baterías, codificadores de voz, …
- Se asume la existencia de *Application Management System* (AMS):
  - dependiente del dispositivo.
  - instala, interacciona con y borra MIDlets.
    - instalación (ej. accediendo a un servidor web vía red inalámbrica)
    - actualización de versiones de MIDlets

Propiedades MIDP

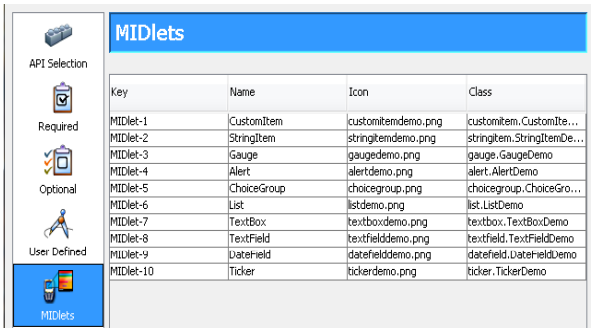
- **MIDP** debe proporcionar al menos las siguientes propiedades
  - **`microedition.locale`**
  - **`microedition.profiles`**
- MIDP 2.0 especifica dos propiedades más dependientes de la implementación
  - **`microedition.commports`**
  - **`microedition.hostname`**

MIDlets

- Un MIDlet es la unidad básica de ejecución en MIDP
  - tiene un ciclo de vida bien definido.
  - da información descriptiva sobre sí mismo.
  - extiende **`javax.microedition.midlet.MIDlet`**
- Existe el concepto de MIDlet permanente:
  - reside, al menos en parte, en memoria no volátil (ROM, EEPROM).
  - puede descargarse de la red y grabarse en memoria persistente.
  - pueden ser ejecutados repetidas veces por el usuario sin necesidad de volver a descargarlos.

MIDlet Suite

- Conjunto de aplicaciones (MIDlets) que comparten recursos en el contexto de una única máquina virtual.



Key	Name	Icon	Class
MIDlet-1	CustomItem	customitemdemo.png	customitem.CustomIte...
MIDlet-2	StringItem	stringitemdemo.png	stringitem.StringItemDe...
MIDlet-3	Gauge	gaugedemo.png	gauge.GaugeDemo
MIDlet-4	Alert	alertdemo.png	alert.AlertDemo
MIDlet-5	ChoiceGroup	choicegroup.png	choicegroup.ChoiceGro...
MIDlet-6	List	listdemo.png	list.ListDemo
MIDlet-7	TextBox	textboxdemo.png	textbox.TextBoxDemo
MIDlet-8	TextField	textfielddemo.png	textfield.TextFieldDemo
MIDlet-9	DateField	datefielddemo.png	datefield.DateFieldDemo
MIDlet-10	Ticker	tickerdemo.png	ticker.TickerDemo

- Aunque sólo desarrollemos un MIDlet se debe empaquetar en un MIDlet Suite.

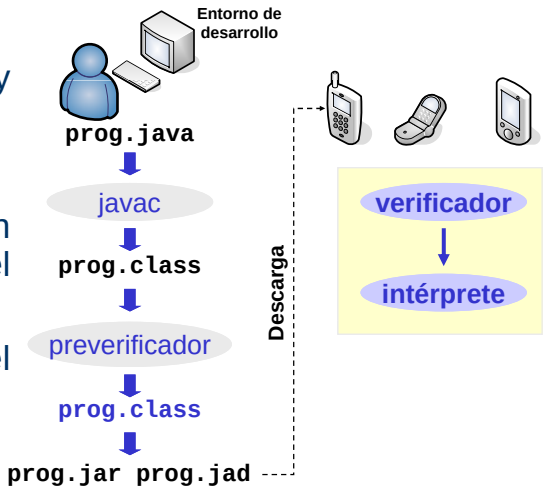
Desarrollo y despliegue de MIDlets

1. Creación (etapas de desarrollo)
2. Publicación
3. Descarga
4. Instalación
5. Ejecución
6. Actualización (gestión de versiones)
7. Borrado

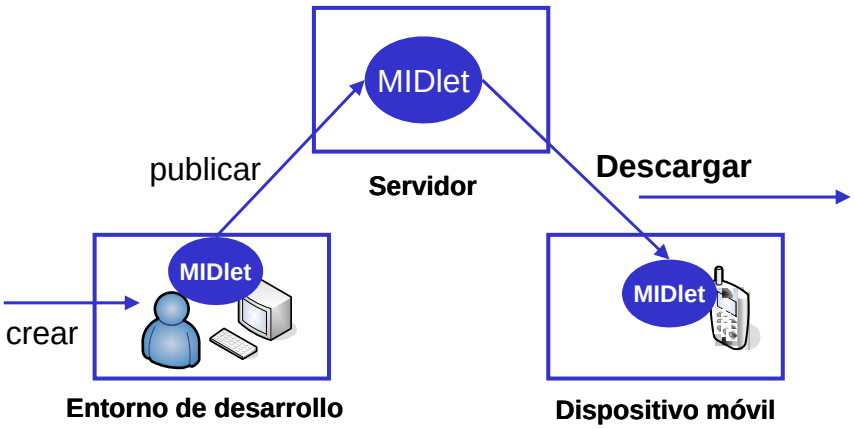
A M S

1. Creación

- Escribir el código y compilar
- Preverificar el código
- Empaquetar en un JAR y crear el descriptor (JAD)
- Ejecutar en el emulador
- Depurar los fallos



2. Publicación



3. Descarga

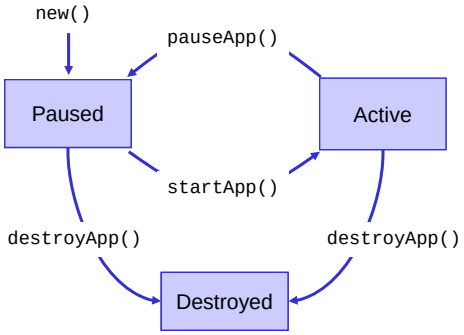
- Gestionada por el *Application Management System (AMS)*
- El dispositivo obtiene el MIDlet de alguna fuente:
  - red inalámbrica (Wi-Fi, Bluetooth, UMTS, GPRS, ...)
  - puerto serie
  - IrDA
  - ...
- Negociación sobre capacidades del dispositivo según los requisitos del MIDlet, coste, ...
- Se descarga el MIDlet a la memoria del dispositivo

4. Instalación

- Gestionado por el *AMS*
  - información al usuario sobre el proceso
- Puede comprobar que el MIDlet no vulnera las políticas de seguridad del móvil
- Puede transformar (convertir) el MIDlet de formato "público" a un formato específico del dispositivo:
  - ejemplo: en PalmOS se transforma a formato PRC.
- El MIDlet queda almacenado en una zona de memoria persistente

5. Ejecución

- El usuario selecciona el MIDlet y lo ejecuta
- En este momento, el MIDlet entra en la VM y se invocan los métodos que gestionan su ciclo de vida:
  - **Paused**: Ha sido creado pero aún no se ha ejecutado y en espera.
  - **Active**: En ejecución.
  - **Destroyed**: Ha liberado recursos, destruido hilos y terminado toda su actividad.



6. Actualización

- Puede publicarse una nueva versión del MIDlet.
- *AMS* debe gestionar la lista de MIDlets instalados y sus versiones
  - puede así actualizar de versiones más antiguas a más recientes del MIDlet
- Los atributos del MIDlet, incluida la versión, están:
  - en el descriptor del MIDlet (JAD).
  - en el manifiesto del MIDlet contenido en el JAR.

7. Borrado

- El AMS debe permitir al usuario eliminar MIDlets
  - Desintalar una aplicación instalada previamente
- Se borra:
  - MIDlet
  - todos los registros en memoria permanente escritos por ese MIDlet
  - los recursos asociados al mismo

JAR y Manifiesto

- Incluye los ficheros de clases y otros recursos asociados al MIDlet, por ejemplo imágenes.
- Fichero JAR puede contener un MIDlet Suite
- El manifiesto está incluido en el JAR y contiene información sobre los contenidos del fichero JAR:

Atributos obligatorios	Atributos opcionales
`MIDlet-Name`	`MIDlet-Description`
`MIDlet-Version`	`MIDlet-Icon`
`MIDlet-Vendor`	`MIDlet-Info-URL`
`MIDlet-<n> (name, icon, class)`	`MIDlet-Data-Size`
`MicroEdition-Profile`	`MIDlet-Permissions`
`MicroEdition-Configuration`	`MIDlet-Permissions-Opt`
	`MIDlet-Push-<n>`

- Otros atributos específicos de la aplicación

Descriptor (JAD)

- Permite que el *AMS* verifique si el MIDlet es indicado antes de descargarlo.
- Es un fichero de texto con extensión **`.jad`**.

Atributos obligatorios	Atributos opcionales
`MIDlet-Name`	`MIDlet-<n> (name, icon, class)`
`MIDlet-Version`	`MicroEdition-Profile`
`MIDlet-Vendor`	`MicroEdition-Configuration`
`MIDlet-Jar-URL`	`MIDlet-Description`
`MIDlet-Jar-Size`	`MIDlet-Icon`
	`MIDlet-Info-URL`
	`MIDlet-Data-Size`
	`MIDlet-Permissions`
	`MIDlet-Permissions-Opt`
	`MIDlet-Push-<n>`
	`MIDlet-Install-Notify`
	`MIDlet-Delete-Notify`
	`MIDlet-Delete-Confirm`

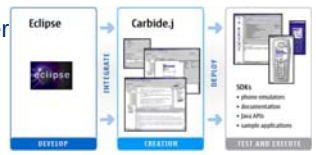
- Puede incluir otros atributos específicos de la aplicación

Ejemplo HolaMundo
Manifiesto y Descriptor

```
MIDlet-Name: HolaMundo
MIDlet-Version: 1.0
MIDlet-Vendor: ITSWC
MIDlet-1:    HolaMundo,    /hola.png,
  uc3m.it.swc.HolaMundo
MIDlet-2:  HolaCanvas,  /canvas.png,
  uc3m.it.swc.HolaCanvas
MicroEdition-Profile: MIDP-2.0
MicroEdition-Configuration: CLDC-1.1
MIDlet-Description: Mi primer MIDlet
MIDlet-Jar-URL: HolaMundo.jar
MIDlet-Jar-Size: 100
```

Desarrollo de aplicaciones

- **SDK de Java 2** ⇨ compilar aplicaciones J2ME
- **Kits de desarrollo**
  - Sun Wireless Toolkit
    - **SDK 3.0 Java ME**
  - Nokia ⇨ carbide.j
    - Eclipse, Sun One Studio, Borland JBuilder
  - Siemens ⇨ Siemens Mobile Toolkit
- **Entornos de desarrollo integrados**
  - **Eclipse**
    - EasyEclipse Mobile Java
    - Plugins: EclipseME, SIPTech J2ME, Wirelesoft VistaMax, …
  - **NetBeans** ⇨ WTK, Plugin J2ME, Mobility Pack 4.1
  - Websphere Studio Device Developer (**WSDD**), IBM

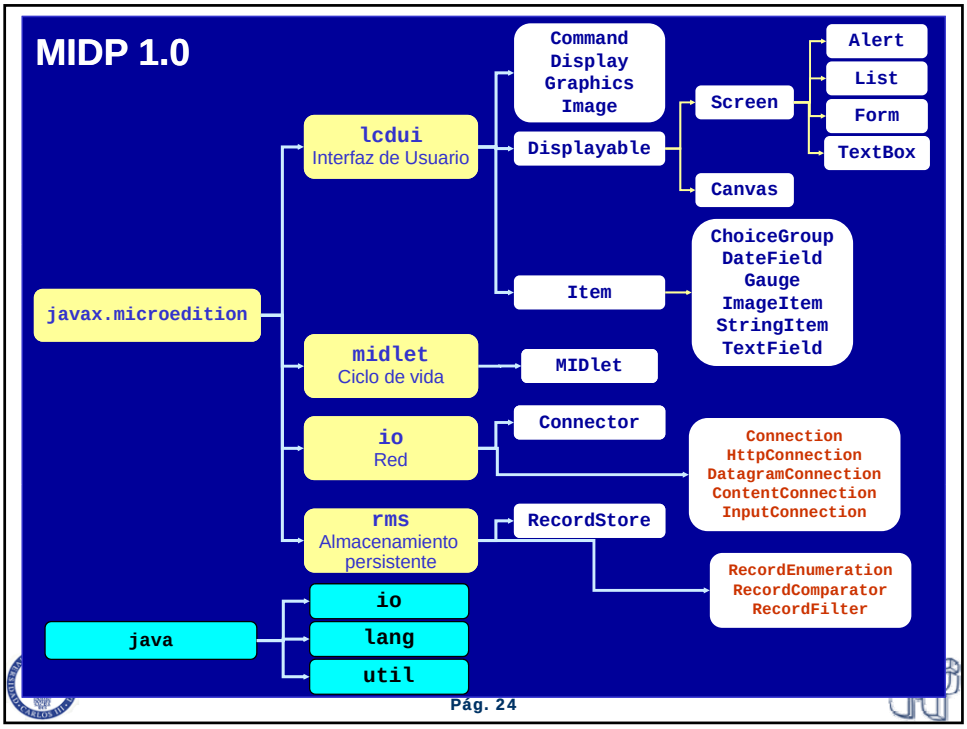


Índice

- Generalidades y conceptos básicos
  - MIDlets y MIDlet Suite
    - desarrollo y despliegue
- **Librerías de MIDP**
- Interfaz de usuario
  - API de alto nivel
  - API de bajo nivel
  - API de juegos
- Almacenamiento persistente
- Conectividad

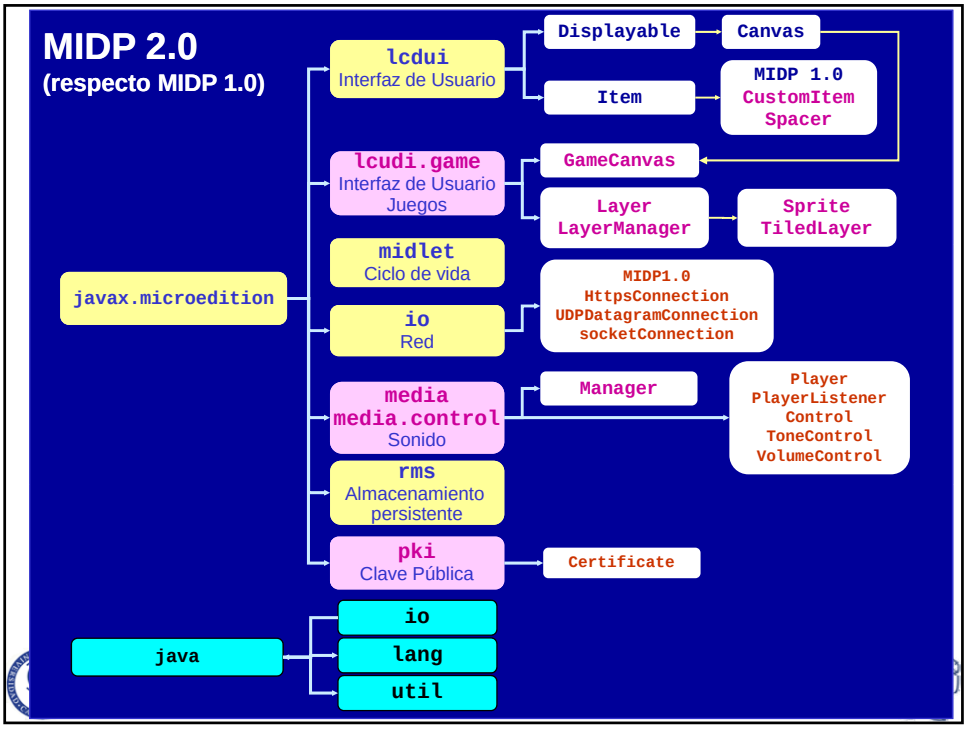
Librerías

- Ciclo de vida de la aplicación (MIDlet):
  - **`javax.microedition.midlet`**
- Interfaz de usuario
  - **`javax.microedition.lcdui`**
- Memoria persistente:
  - **`javax.microedition.rms`**
- Conectividad
  - **`javax.microedition.io`**
- Núcleo
  - **`java.io`**
  - **`java.lang`**
  - **`java.util`**



Librerías añadidas en MIDP 2.0

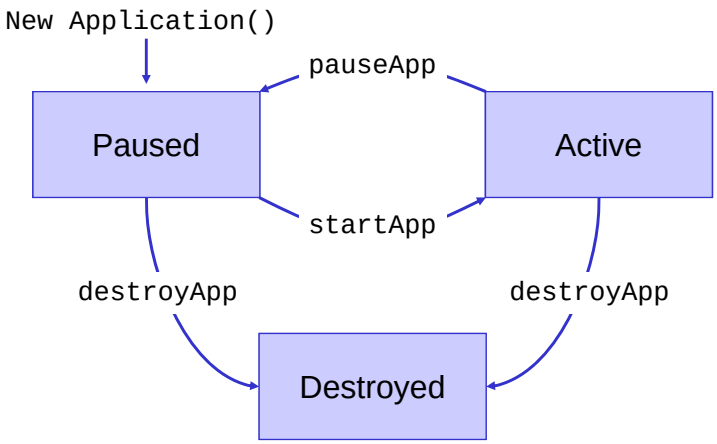
- Interfaz de usuario
  - `javax.microedition.lcdui`
  - `javax.microedition.lcdui.game`
- Seguridad: Clave pública
  - `javax.microedition.pki`
- Sonidos
  - `javax.microedition.media`
  - `javax.microedition.media.control`



MIDlet
`javax.microedition.midlet.MIDlet`

- Clase abstracta base para todos los MIDlets:
  - Constructor: **`protected MIDlet()`**
  - `protected` **`abstract`** `void` **`startApp()`** **`throws MIDletStateChangeException`**
  - `protected` **`abstract`** `void` **`pauseApp()`**
  - `protected` **`abstract`** `void` **`destroyApp(boolean`** `unconditional)` **`throws MIDletStateChangeException`**
  - **`public final void notifyDestroyed()`**
    - Comunica al AMS que el MIDlet ha limpiado la memoria y ha terminado.
  - **`public final void notifyPaused()`**
    - Comunica al AMS que el MIDlet está en pausa.
  - **`public final String getAppProperty(String key)`**
    - Se le llama para obtener las propiedades del MIDlet (descriptor JAD)

Ciclo de vida de un MIDlet



Código ejemplo HolaMundo (I)

```
import javax.microedition.midlet.*;
import javax.microedition.lcdui.*;

public class HolaMundo extends MIDlet
                       implements CommandListener {
  // Componentes de UI del MIDlet
  private Display display;

  private TextBox mainScreen = null;

  private Command exit = new Command("exit",
   Command.EXIT, 2);
```

Implementa **startApp(), pauseApp(), destroyApp()**

Implementa **commandAction(c,s)**

Gestor de la pantalla y dispositivos de entrada. Uno por MIDlet

Permite introducir y editar texto

Botón de comando que permite ejecutar una acción

Código ejemplo HolaMundo (II)

```
// Constructor sin parámetros
public HolaMundo() {
  display = Display.getDisplay(this);

  mainScreen = new TextBox("Text Box", "Hola Mundo",
                           512,0);

  mainScreen.addCommand(exit);

  mainScreen.setCommandListener(this);
}
```

Obtiene una referencia del display del MIDlet

Comando asociado a `mainScreen`

Se establece `mainScreen` como escuchador de "exit"

Código ejemplo HolaMundo (III)

```
// Implementa el método startApp()
public void startApp() {
   display.setCurrent(mainScreen);
 }

// Implementa el método pauseApp()
public void pauseApp() {
 }

// Implementa el método destroyApp()
public void destroyApp(boolean unconditional) {
 }
```

Hace el `TextBox` visible

Ciclo de vida del MIDlet

Código ejemplo HolaMundo (IV)

```
    /*
     * El MIDlet implementa el método escuchador
     * correspondiente del interfaz CommandListener
    */
    public void commandAction(Command c, Displayable s) {
        if (c == exit) {
           destroyApp(true);
           notifyDestroyed();
        }
    }
}
```

Índice

- Generalidades y conceptos básicos
  - MIDlets y MIDlet Suite
    - desarrollo y despliegue
- Librerías de MIDP
- **Interfaz de usuario**
  - API de alto nivel
  - API de bajo nivel
  - API de juegos
- Almacenamiento persistente
- Conectividad

Interfaz de usuario

- API de alto nivel:
  - muy portable.
  - orientada a “screen” y “widget”.
  - las aplicaciones que usan este API deberían funcionar en todos los dispositivos.
  - no hay acceso a todas las funciones del dispositivo.
  - más sencillo y menos potente que AWT.
- API de bajo nivel:
  - primitivas de dibujo.
  - eventos de teclado.
  - menos portabilidad, mejor “experiencia del usuario”.

Interfaz gráfico

- Paquete:
  - **`javax.microedition.lcdui`**
- Clases básicas:
  - **`Displayable`**:
    - Información a ser visualizada.
  - **`Display`**:
    - Selecciona qué objeto **`Displayable`** se muestra al usuario.

Interfaz gráfico
Clase `Displayable`

- Existen tres categorías de objetos `Displayable`:
  - **`Screen`** con estructura predefinida:
    - **`Alert`**, **`List`** o **`TextBox`** (subclases de **`Screen`**)
    - no se pueden enriquecer con nuevos componentes.
  - **`Screen`** genérico:
    - **`Form`** (subclase de **`Screen`**)
    - se pueden llenar con texto, imágenes u otros componentes (objetos `Item`) de interfaz gráfico.
  - **`Canvas`** (API de bajo nivel):
    - Usuario tiene control total sobre los componentes del `display` y puede acceder a eventos de bajo nivel.

Interfaz gráfico
Clase `Display`

- Métodos para controlar la visualización de objetos **`Displayable`** y obtener propiedades del dispositivo
  - color, número de colores, vibración, etc.
- Sólo uno por MIDlet (*singleton*).
- Obtener el objeto `Display`:
  - `static Display getDisplay(MIDlet m)`
- Obtener el **`Displayable`** que se está visualizando:
  - `Displayable getCurrent()`
- Establecer el **`Displayable`** a visualizar
  - `void setCurrent(Displayable nextDisplayable)`
  - `void setCurrent(Alert alert, Displayable nextDisplayable)`

Interfaz gráfico
Gestión de eventos

- Mismo modelo que AWT:
  - Fuentes de eventos y escuchadores (listeners) de evento
- Gestión de eventos en el mismo hilo en el que se produce el evento.
- Eventos de alto nivel:
  - **`CommandAction(Command c, Displayable d)`**
    - `CommandListener`
    - Fuente: `Displayable`
  - **`ItemStateChanged(Item i)`**
    - `ItemStateListener`
    - Fuente: `Form`
    - `Item` interactivos: `Gauge, ChoiceGroup, TextField, …`

Interfaz gráfico
Gestión de eventos (II)

- Eventos de bajo nivel:
  - Relacionados con pulsaciones de teclas, de puntero, …
    - `keyPressed, keyReleased, keyRepetead, pointerPressed, pointerReleased`, …
  - Una aplicación no puede acceder a eventos de bajo nivel a través de un objeto `Displayable` de alto nivel.

Interfaz gráfico
Clase `Command`

- Un objeto **`command`** tiene tres atributos:

  `Command(String label, int Type, int priority)`

  1. *Label*: `String` representando el significado del comando, lo que la aplicación muestra al usuario.
  2. *Type*: `BACK`, `CANCEL`, `HELP`, `EXIT`, `ITEM`, `OK`, `SCREEN` y `STOP`.
  3. *Priority:* Entero que indica la importancia del comando. Mayor cuanto menor sea el número.

- Añadir comando a un `Displayable`:

  `addCommand(Command)`

- Eliminar comando de un `Displayable`:

  `removeCommand(Command)`

Interfaz gráfico
Imágenes

- Clase **`Image:`**
  - Imágenes inmutables:
    - No se pueden modificar.
    - Generadas a partir de un fichero (recurso, descargado,…)
    - Tipo de imágenes en un **`Alert`**, **`List`** o **`Form`**
  - Imágenes mutables:
    - Se pueden modificar.
- ¿Cómo se crean?
  - Inmutables
    - `createImage(String nombre)`
    - `createImage(byte[] data, int offset, int longitud)`
    - `createImage(Image imagen)`
    - `createImage(InputStream stream)`
  - Mutables
    - `createImage(int ancho, int alto)`

API UI de alto nivel
Clase `Screen`

- Elemento funcional de la interfaz de usuario
  - Título, múltiples comandos, y un objeto `Ticker`
- Un objeto de tipo `Ticker` (marquesina) consiste en un texto que se desplaza continuamente a través de la pantalla
  - `Ticker(String texto)`
- Cuatro subclases:
  - `Alert`
  - `List`
  - `TextBox`
  - `Form`

API UI de alto nivel
Clase `Alert`

- Permite visualizar datos (texto, imágenes) durante un cierto tiempo (*timeout*) antes de pasar a otra pantalla
- Constructores:
  - **`Alert(String title)`**
  - **`Alert(String title, String alertText, Image alertImage, AlertType alertType)`**
- *Timeout* en milisegundos:
  - Establecido con el método: **`setTimeout(int time)`**
  - Temporizador infinito (**`Alert.FOREVER`**): cambio de pantalla mediante comando.
- Tipos de alertas
  - **`ALARM`**, **`CONFIRMATION`**, **`ERROR`**, **`INFO`**, **`WARNING`**

API UI de alto nivel
Clase `List`

- Implementa el interfaz **`Choice`**
- Los tipos de listas son definidos en el interfaz
  - Implícitas, exclusivas y de selección múltiple
- Constructores:
  - **`List(String title, int listType):`**
    - Se crea una lista vacía, los elementos de selección se pueden añadir después.
  - **`List(String title, int listType, String[] stringElements, Image[] imageElements:`**
    - Se indican los elementos de la lista y posibles imágenes asociadas a cada elemento.

API UI de alto nivel
Interfaz `Choice`

1. `EXCLUSIVE`
   - Exactamente un elemento seleccionado simultáneamente.
2. `IMPLICIT`
   - Sólo se puede utilizar en listas (**`List`**), donde el elemento que está “enfocado” es el que se selecciona implícitamente.
3. `MULTIPLE`
   - Se puede seleccionar cualquier número de elementos, incluido ninguno, y en cualquier combinación.
4. `POPUP`
   - Tiene exactamente un elemento seleccionado a la vez.
   - No es válido para objetos `List`

API UI de alto nivel
Interfaz `Choice` (II)

Método	Descripción
`int append(String, Image)`	Añade el elemento
`void delete(int)`	Borra el elemento
`String getString(int)`	Obtiene el texto del elemento
`Image getImage(int)`	Obtiene la imagen del elemento
`void insert(int, String, Image)`	Inserta un elemento en la posición previa al elemento indicado
`void set(int, String, Image)`	Modifica el elemento indicado
`void setSelectedFlags(boolean[])`	Modifica el estado de selección de todos los elementos
`void setSelectedIndex(int, boolean)`	Modifica el estado de un elemento concreto
`int size()`	Obtiene el número de elementos

API UI de alto nivel
Ejemplos `List` y `Choice`

Ver código: **`ListDemo.java`** (WTK UIDemo) – **`ListExample.zip`** (proyecto WTK)

API UI de alto nivel
Clase `TextBox`

- Permite al usuario introducir y editar texto
- Constructor
  - **`TextBox(String title, String text, int maxSize, int constraints)`**
- Tamaño máximo:
  - En el constructor y se puede modificar con **`setMaxSize`**.
- Restricciones de entrada:
  - **`ANY`**: cualquier texto
  - **`DECIMAL`**: Números con parte decimal
  - **`EMAILADDR`**: dirección de correo electrónico
  - **`NUMERIC`**: valor entero
  - **`PHONENUMBER`**: número de teléfono
  - **`URL`**: URL

API UI de alto nivel
Clase `TextBox` (II)

- Restricciones de entrada:
  - **`PASSWORD`**: Caracteres introducidos se muestran sustituidos por otro carácter (asteriscos)
  - **`UNEDITABLE`**: El texto no se puede editar
  - **`SENSITIVE`**: El texto no debe almacenarse
  - **`NON_PREDICTIVE`**: Indica el formato de entrada
  - **`INITIAL_CAPS_WORD`**: Mayúscula la letra inicial de cada palabra
  - **`INITIAL_CAPS_SENTENCE`**: Mayúscula la primera letra de cada frase

API UI de alto nivel
Clase `TextBox` (III)

Métodos	Descripción
`void setChars(char[] data, int offset, int legth)`	Establece el contenido como array
`int getChars(char[] data)`	Obtiene el contenido como array
`void setString(String text)`	Establece el contenido como String
`String getString()`	Obtiene el contenido como String
`void insert(char[] data, int offset, int length, int position)` `void insert(String src, int position)`	Modifica el contenido a partir de una posición
`void delete(int offset,int length)`	Borra el contenido

API UI de alto nivel
Clase `Form`

- Contiene un número arbitrario de componentes (items)
  - El dispositivo controla la posición y el desplazamiento
- Constructores:
  - `Form(String title)`
  - `Form(String title, Item[] items)`
- Métodos:
  - `int append(Image img)`
  - `int append(Item item)`
  - `int append(String str)`
  - `void delete(int itemNum)`
  - `void insert(int itemNum, Item item)`
  - `void set(int itemNum, Item item)`
  - `Item get(int itemNum)`
- Un `Item` sólo puede colocarse en un `Form`

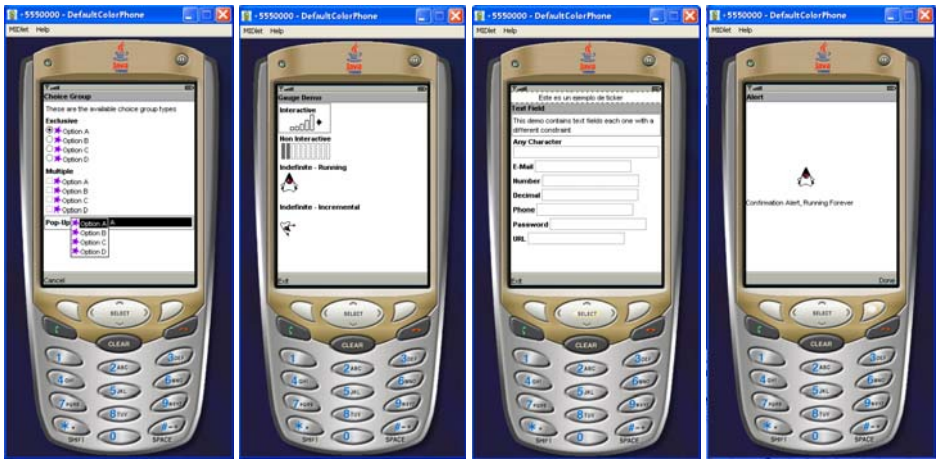
API UI de alto nivel
Clase `Item`

- Conjunto de elementos que pueden añadirse a un `Form`, con una etiqueta asociada.
- Superclase de:
  - **`StringItem`**:
    - Visualizar texto (sólo las aplicaciones interaccionan, no el usuario)
  - **`ImageItem`**:
    - Con parámetro “layout” para alineación
  - **`TextField`**:
    - Editar texto
  - **`DataField`**:
    - Visualizar fechas y horas

API UI de alto nivel
Clase `Item` (II)

- Superclase de:
  - **`ChoiceGroup`**:
    - `Item` que implementa la interfaz `Choice` (no implícita)
    - Parecido a **`List`**.
  - **`Gauge`**:
    - Diagrama de barras
  - **`Spacer:`**
    - Ajustar el espacio entre componentes visibles
  - **`CustomItem:`**
    - Clase abstracta que permite definir nuevos tipos de Items.
    - MIDP 2.0.

API UI de alto nivel
Ejemplos `Form` e `Item`



Ver código: **`UIExample.zip`** (proyecto ejemplo WTK 2.2)