# T1-Interfaces.pdf

**bloodyraintatii**

**Desarrollo de interfaces**

**1º Desarrollo de Aplicaciones Multiplataforma**

**Estudios España**

Accede al documento original

# Desarrollo de Interfaces

## Tema 2: Clases y componentes

b) Se han utilizado las funciones del editor para ubicar los componentes del interfaz.

### Área de diseño

→ Partes principales de la vista de diseño.
- Zona de diseño → donde irán los componentes de la interfaz.
- Palette → elementos usados para la implementación de la interfaz.
- Components → lista de elementos insertados en la zona de diseño.
- Propiedades → propiedades de los elementos que definen su apariencia entre otras cosas.

→ Para insertar un elemento pulsamos sobre él y lo llevamos a la zona de diseño.
→ Para eliminar un elemento le hacemos click derecho y le damos a Borrar.

### JFrame y JPanel

**JFrame**
→ Ventana de la interfaz.
→ Necesita un constructor para crear una instancia de la interfaz → JFrame() / JFrame(String nombreVentana).
→ Se establecen las dimensiones, acción de cierre y visibilidad.

**JPanel**
→ Panel o lienzo. Es el bloque invisible (en principio) que se sitúa sobre una ventana y donde se ubican más elementos.
→ La distribución de estos paneles constituye un sistema de capas o Layout.

**Resto de componentes**
→ Se colocan dentro del lienzo → panel.add(button);

### JDialog

→ Ventana de tipo secundario que se abrirá por encima de su padre (un JFrame u otro JDialog).
→ Creación → File > New > Other > JDialog.
→ Diálogos modales → no permiten que otras ventanas de diálogo se abran hasta que la que está abierta se haya cerrado completamente → JDialog ventana = new JDialog(f, “Dialog”, **true**);

## Componentes

**JButton**
→ Botón → JButton button = new JButton("Soy un botón");
→ Propiedades → background, enabled, font, foreground (color del texto), horizontalAlignment, verticalAlignment, text, icon.

**JLabel**
→ Etiqueta que puede contener textos, imágenes o iconos.
→ Propiedades → background (color del texto cuando está habilitado), enabled, font, foreground (color del texto cuando no está habilitado), horizontalAlignment, verticalAlignment, text, icon.

**JTextField**
→ Contenedor de una línea de texto.
→ Propiedades → background, columns, enabled, editable, font, foreground (color del texto), horizontalAlignment, text.

**JCheckBox**
→ Elemento de tipo casilla que pueden ser marcados ☐.
→ Atributo "selected" → dará true si está marcado y false si no lo está.

**JRadioButton**
→ Elemento que indica varias opciones de las que solo se puede elegir una.
→ Se añaden todos los JRadioButton a un ButtonGroup para que solo se pueda elegir uno
→ ButtonGroup bg = new ButtonGroup(); bg.add(rb1); bg.add(rb2);

**JComboBox**
→ Menú desplegable con varios valores.
→ Propiedad "maximumRowCount" → número máximo de elementos mostrados.
→ Propiedad "selectedIndex" → indica qué valor se mostrará por defecto.

## Layout manager

**Flow Layout**
→ Sitúa los elementos uno al lado del otro en la misma fila.
→ setAlignment → dar valor al tipo de alineación.
→ setVgap / setHgap → distancia de separación entre los elementos.

**Grid Layout**
→ Sitúa los componentes siguiendo un patrón de columnas y filas → GridLayout(int numFilas, int numCol);
→ También se puede modificar la distancia de separación entre componentes.
→ column / row → definen el número de columnas y filas.

**Border Layout**
→ Sitúa los elementos en los extremos del panel y en el centro.
→ Se debe poner NORTH / SOUTH / EAST / WEST / CENTER al agregar el elemento →
contentPane.add(button, BorderLayout.NORTH);

**GridBag Layout**
→ Cada componente tendrá asociado un objeto de tipo GridBagConstraints.
→ Es más flexible que Grid Layout.


# Tema 3: Generación de interfaces a partir de documentos XML

a) Se han reconocido las ventajas de generar interfaces de usuario a partir de su descripción en XML.

## SVG

→ Scalable Vector Graphics / Gráficos Vectoriales Escalables.
→ Permite representar elementos geométricos vectoriales, imágenes de mapa de bits y texto.

**Crear formas geométricas con XML**

- Cuadrado → `<svg width="60" height="60"> <rect x="0" y="0" width="60" height="60" fill="red"/> </svg>`
- Círculo → `<svg height="100" width="100"> <circle cx="50" cy="50" r="40" stroke="black" stroke-width="3" fill="cyan" /> </svg>`
- Elipse → `<svg width="60" height="60"> <ellipse cx="30" cy="30" rx="20" ry="16" fill="orange"/> </svg>`
- Triángulo → `<svg width="60" height="60"> <polygon fill="green" stroke="black" stroke-width="2" points="05,30 15,10 25,30"/></svg>`

## Generación de código para diferentes plataformas

→ Los datos XML pueden ser leídos por diferentes plataformas, lo que reduce problemas de incompatibilidad. Cualquier programa diseñado para leer XML puede leer y procesar los datos XML independientemente del hardware o SO.
→ XML se ha convertido en una de las tecnologías más usadas como base para el almacenaje de contenidos, como modelo de representación de metadatos y como medio de intercambio de contenidos:

- XML para el almacenaje de contenidos → bases de datos XML nativas que almacenan y gestionan documentos XML directamente.
- XML como medio de intercambio de contenidos → se pueden procesar para múltiples fines: integración en una bbdd, visualización como parte de un sitio web o mensajes entre aplicaciones.
- XmL para la representación de metadatos → lo más importante es el sistema de indexación y recuperación para poder discriminar dentro del contenido los elementos o atributos que se quieren recuperar.

# Tema 5: Explotación de componentes visuales

d) Se han determinado los eventos a los que debe responder el componente y se les han asociado las acciones correspondientes.

## Eventos: clases

→ La programación basada en eventos se puede dividir en la detección de los eventos y las acciones asociadas a su tratamiento.
→ Eventos internos → producido por el sistema.
→ Eventos externos → producidos por el usuario, normalmente a través del teclado o ratón.

- EventObject → clase principal de la que derivan todos los eventos.
- MouseEvent → eventos del ratón.
- ComponentEvent → eventos de un componente; p.ej. cambio de tamaño.
- ContainerEvent → evento producido al añadir o eliminar un elemento de un Container.
- WindowEvent → evento de una ventana al sufrir variación; p.ej. apertura, cierre, cambio de tamaño.
- ActionEvent → evento producido al detectarse acción sobre un componente.

## Eventos: componentes

→ Los tipos de componentes tienen diferentes eventos asociados.

- JTextField → ActionEvent → Enter tras completar un campo de texto.
- JButton → ActionEvent → pulsar en un botón.
- JComboBox → ActionEvent ItemEvent → se selecciona uno de los valores.
- JCheckBox → ActionEvent ItemEvent → se marca una celda de selección.
- JTextComponent → TextEvent → cambio en el texto.
- JScrollBar → AdjustmentEvent → se hace scroll (barra de desplazamiento).

## Listeners

**KeyListener**
→ Al pulsar cualquier tecla.
→ Se implementan los eventos ActionEvent.

- keyPressed → se pulsa la tecla.
- keyTyped → se pulsa y suelta la tecla.
- keyReleased → se suelta la tecla.

**ActionListener**
→ Se pulsa un componente.
→ Implementan los eventos ActionEvent.

- JButton → se pulsa o se da Enter teniendo el foco en el botón.
- JTextField → se da Enter al tener el foco en la caja de texto.

# POV: quedas con alguien

## y no se parece a sus fotos.

Si te pasa con un pedido, el seguro de compra online* de tu **Cuenta NoCuenta** de ING te cubre. **¡Y es gratis!**

**Saber más**

*Compras superiores a 30 €. Más info en ing.es

- JMenuItem → se selecciona una opción del menú.
- JList → se hace doble click en un elemento de la lista.

**MouseListener**
→ Se realizan acciones con el ratón.
- mouseClicked → pulsar y soltar sobre un componente.
- mouseExited → salir de un componente.
- mousePressed → presionar sobre un componente.
- mouseReleased → soltar el ratón sobre un componente.
- mouseEntered → se entra a un componente.

**FocusListener**
→ Ocurre cuando un elemento está seleccionado (tiene el foco) o deja de estarlo.
→ Implementa objetos de la clase FocusEvent.

**MouseMotionListener**
→ Detecta el movimiento del ratón.
- mouseMoved → se mueve el ratón sobre un componente.
- mouseDragged → se arrastra el ratón pulsado sobre un componente.

## Eventos: métodos

→ Será necesario ejecutar un método u otro dependiendo del tipo de evento asociado.

| Listener | Métodos | |
|---|---|---|
| ActionListener | public void actionPerformed(ActionEvent e) | |
| KeyListener | keyPressed | public void keyPressed(KeyEvent e) |
| | keyTyped | public void keyTyped(KeyEvent e) |
| | keyRelease | public void keyReleased(KeyEvent e) |
| FocusListener | Obtención del foco | public void focusGained(FocusEvent e) |
| | Pérdida del foco | public void lostGained(FocusEvent e) |
| MouseListener | mouseClicked | public void mouseClicked(MouseEvent e) |
| | mouseExited | public void mouseExited(MouseEvent e) |
| | mousePressed | public void mousePressed(MouseEvent e) |
| | mouseReleased | public void mouseReleased(MouseEvent e) |
| | mouseEntered | public void mouseEntered(MouseEvent e) |
| MouseMotionListener | mouseMoved | public void mouseMoved(MouseEvent e) |
| | mouseDragged | public void mouseDragged(MouseEvent e) |

## Asociación de acciones a eventos

→ Secuencia de pasos comunes para asociar una acción a un evento:

- Creación del listener:
  - normal → nombreComponente.addtipoEventoListener(new tipoEventoListener() {…}
  - modelando primero el tratamiento del evento y asociándolo al componente sobre el que actúa → tipoEventoListener nombreEvento = new tipoEventoListener (){…} nombreComponente.addtipoEventoListener(nombreEvento);
- Asociación de la acción al evento → public void métodoDeEvento(TipoEvento e){…}

→ Los métodos son relativos a cada evento.

# Tema 6

a) Se han creado menús que se ajustan a los estándares.
g) Se ha verificado que los mensajes generados por la aplicación son adecuados en extensión y claridad.
h) Se han realizado pruebas para evaluar la usabilidad de la aplicación.

## Usabilidad

→ Usabilidad / usability → facilidad o dificultad de uso de un entorno de software que implique la interacción con el usuario.
→ Parámetros por los que se define la usabilidad:

- Eficiencia de uso → tiempo requerido para completar una acción. P.ej. encontrar botones.
- Facilidad de aprendizaje → tiempo requerido para conocer el funcionamiento de la aplicación.
- Retención del tiempo → el tiempo de aprendizaje que precise el usuario debe ser menor que el de la primera vez.
- Satisfacción → grado de satisfacción del usuario con respecto al sistema. Es subjetivo.
- Tasa de error → el número de errores de los usuarios debe ser el menor posible, sobre todo si es por la complejidad de la herramienta.

→ Principios para el diseño de sistemas interactivos:

- Conocer al usuario → implementar según los “usuarios tipo”.
- Minimizar la memorización → sustituir las entradas de datos por la selección de ítems.
- Optimizar las operaciones mediante la rápida ejecución de operaciones comunes → reorganizar la estructura de la información según el uso del sistema.
- Facilitar buenos mensajes de error, crear diseños que eviten los errores más comunes → haciendo posible deshacer acciones realizadas y garantizando la integridad del sistema en caso de errores.

## Normas y estándares de usabilidad

→ ISO (International Organization for Standardization) → crea las normas y estándares para asegurar que servicios y productos tengan ciertos niveles de calidad, eficiencia y seguridad.
→ Usabilidad según ISO → capacidad de un producto para ser entendido, aprendido, usado y resultar atractivo para el usuario cuando se usa bajo ciertas condiciones.
→ Principales normas ISO:

- Estándar de calidad del producto → ISO/IEC 9126-1. Relativa al uso.
- Guía de usabilidad → ISO/IEC 9241. Relativa al uso.
- Guía de interfaz multimedia → IES TR 61997. Relativa a la interfaz e interacción.
- Estándar de ergonomía → ISO/IEC 14915. Relativa a la interfaz e interacción.

## Medidas de usabilidad en aplicaciones

→ Las medidas de usabilidad son una herramienta clave para evaluar la usabilidad.
→ Los tests de usabilidad evalúan la facilidad de uso de un usuario y si la funcionalidad desarrollada cumple con la finalidad de la aplicación.

**Satisfacción**
→ El nivel de satisfacción es clave para la evaluación de la aplicación. Un bajo nivel supondrá la modificación del sistema.
→ Métricas → calificación de satisfacción del usuario sobre la aplicación, frecuencia de reutilización de la aplicación, calificación relativa a la facilidad de aprendizaje, y medida de uso voluntario de la aplicación.

**Efectividad**
→ La efectividad determina el grado de éxito de una aplicación a la hora de cumplir con su finalidad. Está ligado a la facilidad de aprendizaje.
→ Métricas → cantidad de tareas relevantes completadas en cada intento; nº de acceso a la documentación, soporte y ayuda; cantidad de funciones aprendidas; nº de usuarios capaces de aprender las características del producto; cantidad y tipos de errores tolerados por los usuarios; cantidad o porcentaje de palabras leídas correctamente.

**Eficiencia**
→ La eficiencia se define según el tiempo necesario para completar una tarea.
→ Métricas → tiempo productivo; tiempo para aprender el funcionamiento; tiempo requerido en el 1er intento para completar la funcionalidad evaluada; eficiencia relativa al 1er intento; errores persistentes; tiempo necesario para aprender de nuevo la funcionalidad del producto pasado un tiempo desde su anterior uso.

## Diseño y realización de pruebas de usabilidad

→ El desarrollo de pruebas de usabilidad requiere de un algoritmo.
→ Cuando se completa el primer diseño y se realizan las pruebas, no se para ahí, sino que el proceso de retroalimentación se extiende todo lo necesario.

**Pruebas con expertos**
→ Evaluación heurística o método de inspección → analizan e identifican los problemas existentes o posibles antes de llevar la aplicación a producción.
→ Se divide en dos partes:

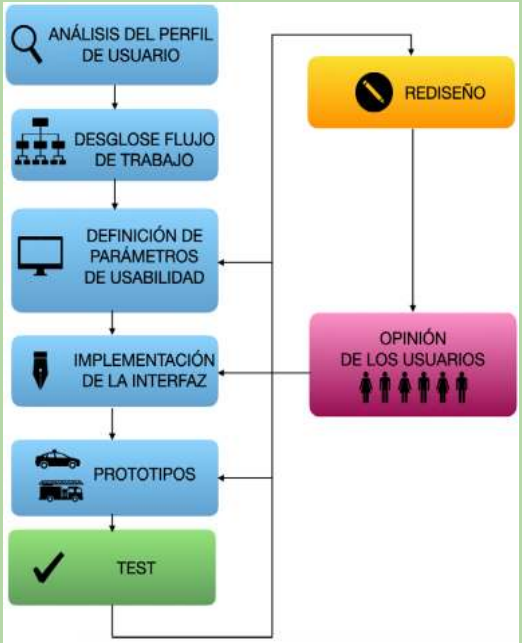

*Fig. 6. Diagrama de test de usabilidad.*





- Evaluación detalle → se analiza de forma exhaustiva cada pantalla, botón, etc.
- Evaluación de alto nivel → se analiza el funcionamiento de forma general.

→ En torno a 4 expertos trabajan independientemente. Al final se hará un informe conjunto.
→ Estas pruebas se hacen en cualquier momento del proceso de desarrollo; es aconsejable que se hagan antes de las pruebas con usuarios.

**Pruebas con usuarios**
→ Se basan en el análisis y evaluación de un software mediante un grupo de usuarios reales que pueden detectar errores.
→ Son al menos 15 usuarios con los perfiles a los que está dirigida la aplicación.
→ Pruebas individuales teniendo en cuenta todas las observaciones.
→ Algunos criterios de diseño: pruebas razonables, pruebas específicas, pruebas factibles, tiempo de realización razonable.

**Tipos de test de usabilidad**
→ Tipos de test de usabilidad:

- Test de uso pautado → un responsable monitorea las pruebas hechas a un prototipo muy cercano a la versión real.
- Test de uso descontextualizado → como el anterior pero sobre un prototipo real.
- Uso natural → pruebas sobre la versión final sin moderador que paute las acciones.
- Híbridos → combinación de las anteriores.

**Diseño de pruebas con usuarios**
→ Pautas para el diseño de pruebas de usabilidad:

- Se definen los objetivos de las pruebas.
- Se diseña el formato y tipo de datos para analizar en el estudio.
- Se realiza el diseño de las pruebas.
- Se determina el nº de personas que participarán en el estudio.
- Se eligen las métricas que se van a recoger por los tests.
- Se implementa el plan de test, con la guía de las pruebas, el tiempo destinado a ellas, etc.

# Tema 7: Usabilidad II: Pautas de diseño

todo

## Pautas de diseño de la estructura de la interfaz de usuario

→ A la hora de hacer diseños, nos debemos enfocar en el punto focal, la estructura y consistencia de ventanas, la relación entre elementos, la legibilidad y flujo entre elementos, y los atajos de teclado.

**Atajos de teclado**
→ Permiten reducir el tiempo de acceso y puesta en ejecución de determinadas acciones. Por eso, los atajos suelen asociarse a las tareas más frecuentes.

**Menús**
→ Deben permitir la correcta navegación de la aplicación, mostrando y permitiendo seleccionar las diferentes acciones del menú.

- Se debe mostrar el título del menú.
- Se muestran las opciones y la acción asociada.

→ Debe ser posible una navegación rápida e intuitiva.

- No se aconseja el uso de menús en cascada.
- No se aconseja que los menús tengan muchas acciones → entre 7 y 10.
- Se aconseja que los elementos aparezcan también en otro sitio.

**Ventanas y cuadros de diálogo**
→ El diseño de ventanas debe estar correctamente escogido y desarrollado, así como el número de ventanas totales o el sistema de apertura y cierre de estas.
→ Los usuarios deben poder abrirlas, cerrarlas, modificar su tamaño… si no, debe existir justificación lógica.
→ Es un mal diseño si se despliegan ventanas constantemente pero no se cierran.
→ Los cuadros de diálogo permiten una comunicación activa entre la interfaz y el usuario. Los mensajes deben ser activos y positivos, y adaptativos a las variantes culturales. Deben describir claramente el mensaje sin dar por sentada información.

**Iconos y colores**
→ El aspecto de una interfaz se centra en: colores, fuente, iconos y distribución de los elementos. Una mala elección puede llevar al fracaso.

- Iconos → representación de la acción que enlaza. Debe ser representativa y sencilla.
- Colores → se deben escoger adecuadamente para la comunicación del mensaje. Pueden contribuir a poner el foco en los elementos más importantes, aumentando la eficiencia y velocidad de uso. No se debe abusar del número de colores.

*Fig. 6. Ejemplos de psicología del color aplicados al diseño de interfaces.*

- Fuentes → el uso de fuentes familiares mejora la calidad de la lectura. La fuente debe seleccionarse según la legibilidad y adaptación a la pantalla. Los criterios más importantes son: tamaño de la fuente, color de la fuente y estilo.
- Distribución de los elementos → las ventanas o cuadros de diálogo también requieren atención para no confundir a los usuarios sino hacer la experiencia de uso más intuitiva. Se evitan el uso de elementos y ventanas superpuestas. Los elementos se disponen de manera que facilite el seguimiento y lectura de estos.

## Pautas de diseño de elementos interactivos del interfaz de usuario

→ Elementos interactivos → son los que permiten la comunicación activa entre interfaz y usuario.

**Cuadros de texto y etiquetas**
→ Las cajas para introducir texto deben incorporar un texto explicativo del input que se espera y deben estar ajustadas a la ventana.

**Botones, checkBox o radioButton**
→ Permiten escoger valor o valores y enviarlos a la aplicación. Los títulos deben ser intuitivos. Las acciones esperadas deben ser comprensibles para el usuario. Las opciones deben ser fácilmente distinguibles.

**Menús desplegables**
→ Permiten escoger valor o valores y enviarlos a la aplicación. Deben cumplir pautas como no pasar de 10 elementos en total.

## Pautas de diseño de la presentación de datos

**Tablas**
→ Suelen mostrar info estructurada pero no se debe abusar de ellas.
- Las etiquetas deben ser claras y concisas.
- El título de la tabla debe aparecer y no debe ocupar más de dos líneas de texto.
- Se aconseja usar encabezados de filas o columnas para resumir su contenido.
- La información debe mostrarse claramente.

**Gráficos**
→ Su uso también debe ser balanceado.
- El tamaño debe adecuarse a la pantalla.
- No se debe abusar de su uso.
- Se aconseja usar pocos y que aporten un valor añadido a la aplicación.

## Pautas de diseño de la secuencia de control de la aplicación

→ El diseño secuencial de acciones debe partir de lo general a lo particular.
→ Hay que desarrollar varios prototipos sobre el funcionamiento que se quiere conseguir y optimizar el número de pasos para completar la acción.

- Se debe diseñar y establecer claramente el objetivo de cada elemento.
- Se debe establecer la consecución de cada objetivo con una secuencia de control válida.
- Se debe mostrar y documentar la secuencia establecida para que el usuario pueda implementarla sin problemas.
- Se debe usar la regla de los tres clicks si es posible → que sea posible llegar a cualquier objetivo en máximo 3 clicks.

## Pautas de diseño para el aseguramiento de la información

→ Hay que asegurar la consistencia de la información. Esto implica:

- Integridad de la información.
- Disponibilidad de los datos cuando se requieren.
- Confidencialidad de la info bajo el diseño de los procesos de autenticación oportunos.

→ El aseguramiento de la información afecta a datos, procesos, comportamiento y sistema de gestión de la empresa.

## Pautas de diseño para aplicaciones multimedia

**Imágenes**
→ Uso balanceado.
→ Deben tener buena calidad.

- El tamaño debe ajustarse.
- No conviene abusar de las imágenes.
- Deben aportar un valor añadido, usando diagramas de uso o funcionamiento útiles para la utilización de la herramienta.

**Animaciones**
→ Son vídeos pequeños compuestos por una breve secuencia de imágenes en movimiento.
→ No se debe sobrecargar la aplicación, ya que pueden incluso restarle valor.

**Vídeos**
→ Se usan para mostrar visualmente acciones, procesos o productos.

- El tamaño debe ajustarse.
- No conviene abusar.
- Se deben añadir los elementos necesarios para que se pueda controlar la reproducción.

interfaces
---t2-1b
b) Se han utilizado las funciones del editor para ubicar los componentes del interfaz.
---t3-2a
a) Se han reconocido las ventajas de generar interfaces de usuario a partir de su descripción en XML.
---t5-3d
d) Se han determinado los eventos a los que debe responder el componente y se les han asociado las
acciones correspondientes.
---t6-4a, 4g, 4h
a) Se han creado menús que se ajustan a los estándares
g) Se ha verificado que los mensajes generados por la aplicación son adecuados en extensión y claridad.
h) Se han realizado pruebas para evaluar la usabilidad de la aplicación.
---t7-todo