1. Componentes visuales

1.1. Introducción a los componentes visuales

El nombre con el que se conoce el conjunto de elementos visuales de una interfaz gráfica junto con la propia interfaz, en un entorno de ordenador personal, es el de WIMP (windows, icon , menu , pointer device).

> **Android**
>
> Es un sistema operativo para dispositivos móviles desarrollado por Google basado en el núcleo (kernel) de Linux.

Se conoce como estilo de interacción post-WIMP a las interfaces de los dispositivos móviles y demás.

1.1.1. Componentes visuales: concepto y características

Las interfaces gráficas de usuario utilizan elementos visuales para representar la información y los datos a mostrar.

> **Clase**
>
> Una clase es un conjunto de propiedades y métodos relacionados con su propio significado, en un contexto de programación orientada a objetos.

Un **componente visual** es una clase de uso específico, lista para ser arrastrada a un formulario, que se podrá configurar o utilizar de forma visual, desde el entorno de desarrollo. Son elementos visuales que permiten la gestión y representación de información.

La diferencia principal, respecto a una clase normal, es que la mayor parte del trabajo de configuración del elemento puede realizarse de forma visual, con el ratón, y ajustando las opciones y propiedades que se ofrecen en su entorno.

En definitiva, se pueden modificar las propiedades de los componentes y construir los gestores de eventos a los que éstos pueden responder.

Se utilizan IDEs para realizar el desarrollo de las aplicaciones teniendo a disposición un multitud de componentes visuales.

Algunos ejemplos de componentes más simples son:

- Botones
- Listas desplegables
- Barras de progreso
- Etiquetas
- Radio buttons
- …

Algunos ejemplos de componentes más complejos son:

- Menús
- Tablas
- Árboles
- Cuadros de diálogo
- …

Los componentes tienen, entre otros, las siguientes características conceptuales y de creación:

- Permiten la interacción con otros componentes.
- Son una unidad de software compilada reutilizable, con una interfaz bien definida.
- Proporcionan el control sobre determinados recursos externos en tiempo de diseño.
- Se distribuyen en un único paquete que contiene en sí mismo todo lo necesario para su funcionamiento.
- Pueden haberse desarrollado en cualquier lenguaje de programación.
- Pueden utilizarse para el desarrollo de aplicaciones en cualquier lenguaje de programación.
- Pueden ser visibles o no visibles para el usuario en tiempo de ejecución, aunque siempre serán visibles para el desarrollador en tiempo de ejecución.

Respecto a la funcionalidad, los componentes también disponen de las siguientes características:

- Los componentes instalados son accesibles en el entorno de desarrollo por medio de una paleta de componentes y pueden arrastrarse a la aplicación en la que se está trabajando.
- Siempre que sean visibles para el usuario ,es posible manipularlos de forma interactiva.
- Cada componente expone públicamente un conjunto de propiedades que indican los valores de sus características, es decir, su estado interno.
- Cada componente genera una serie de eventos que se pueden capturar.

1.1.2. Propiedades y atributos de los componentes visuales

Las propiedades y atributos de un elemento visual muestran todas las posibles características de ese componente, desde las más sencillas hasta las más complejas o elaboradas, como la forma de la que actuará cuando se haga clic con el ratón encima o cómo debe comportarse el componente cuando suceda un evento vinculado con él.

Los atributos o propiedades de un componente pueden modificarse mediante código, o mediante la ventana de propiedades.

Además de sus propiedades, los componentes también dispondrán de métodos asociados.

1.1.3. Eventos vinculados

Durante el tiempo de ejecución de una aplicación se podrán ejecutar por parte del usuario o podrán suceder de forma automática diferentes sucesos o eventos.

Estos sucesos podrán ser síncronos o asíncronos.

Los lenguajes de programación permiten ejecutar partes de código como respuesta a acciones o eventos que, normalmente, serán asíncronos, durante la ejecución de la aplicación.

Un ratón o un teclado o una pantalla táctil, pueden ser ejemplos de elementos capaces de desencadenar un evento.

Dentro de sus propiedades, los componentes deben disponer de la posibilidad de vincular partes de código a que se produzcan algunos eventos o, también, la posibilidad de no vincular nada a ese evento.

Algunos ejemplos de eventos relacionados con los ratones para un componente cualquiera podrían ser:

- Click . Se podrá indicar la parte de código a ejecutar al hacer clic con el ratón (botón izquierdo) sobre el componente.
- MouseOver . Este evento indica que el ratón está encima de un componente determinado.
- MouseMove . Este evento tiene lugar cuando el usuario mueve el ratón por encima de un componente.
- MouseDown . Cuando el usuario hace clic con el ratón sobre cualquier botón, derecho o izquierdo.
- MouseOut . Este evento indicará que el puntero del ratón ha salido de una cierta zona de la interfaz o que deja de estar sobre un componente.
- MouseClicked . Es otra forma de llamar al evento Click (un clic con el ratón sobre un componente).
- MouseDouble - Clicked . Este componente se produce cuando se hace clic dos veces de forma consecutiva sobre un componente con el ratón.
- MouseEntered . Equivalente al MouseOver . Por ejemplo, al pasar sobre un componente con un ratón, el texto que éste contenga modificará su color.
- MouseExited . Equivalente al MouseOut . Por ejemplo, al salir con el ratón de un componente, el color del texto que éste contenga volverá al color original.

1.1.4. Tipo de componentes: clasificaciones

- Visuales / no visuales
- Si son controles
- Si son contenedores
- Componentes estructurales
- Componentes de interacción
- Componentes avanzados para la web 2.0

La clasificación de los componentes en visuales y no visuales se basa en que el componente sea visible o no para el usuario final de la aplicación.

Un componente es visual cuando tiene una representación gráfica en tiempo de diseño y ejecución y se dice no visual de lo contrario.

Otra clasificación posible de los componentes será según su funcionalidad.

Los componentes de tipo contenedor son componentes que agruparán otros.

Los componentes estructurales son los que sirven para crear la estructura estática con la que el usuario podrá interactuar.

Los componentes de interacción permiten a los desarrolladores de aplicaciones intercambiar acciones con los usuarios, ofreciéndoles informaciones y recogiendo sus respuestas.

Un ejemplo de componentes de interacción son los componentes de tipo control (conocidos directamente como controles).
Los controles son objetos que proporcionan una interacción entre la aplicación y los usuarios y el intercambio de información.

Ejemplos de componentes de tipo control son:

- Cuadro de lista o list box
- Combo box
- Icono o icon
- Tree view
- Botón o button
- Casilla de selección o check box
- Radio button
- Lista desplegable o drop-down list
- Menú o menú
- Barra de menú o menú bar
- Barra de herramientas o toolbar
- Grid view

Son los denominados componentes para la web 2.0 . Un ejemplo de componentes de estas características les ofrece la herramienta jQuery, entorno de trabajo (framework) que ofrece muchas oportunidades de este tipo de componentes.

1.1.5 Algunos ejemplos de componentes

- Labels (etiquetas) , que ofrecen la posibilidad de ubicar en el formulario una etiqueta con un texto determinado, que no podrá ser manipulado por el usuario de la aplicación.
- Buttons (botones) , que ejecutarán una funcionalidad cuando se haga un clic o doble clic.

- Picture box (imágenes) , que permiten seleccionar una imagen o un gráfico de un archivo previamente existente.
- Progress bar (barra de progreso) , que ofrece la progresión en la ejecución de una funcionalidad determinada que se está ejecutado.
- Radio buttons (botones de opción) , que son un conjunto de opciones agrupadas que ofrecen diferentes alternativas al usuario para poder escoger una.
- Check box (casilla de selección) , que son un conjunto de opciones con un espacio en forma de cuadrado a la derecha para poder seleccionarlas.
- List box (cuadro de lista) , que muestra una lista de opciones en formato de lista.
- Text box (cuadro de texto) , que muestra información en formato de texto escrito en tiempo de diseño y recoge información introducida por el usuario en tiempo de ejecución.
- Combo box , que ofrece una lista de opciones entre las que se podrá seleccionar una o introducir un texto en el cuadro de edición.

1.1.6. Plataformas rich client y thin client (cliente ligero)

Otra forma de clasificar los componentes visuales es teniendo en cuenta para qué tipo de entorno se han desarrollado.

Existen las plataformas llamadas **thin client**, en las que los programas y aplicaciones desarrolladas dependen en gran medida de otros programas para su ejecución. Por otro lado, existen las plataformas llamadas **rich client**, en las que los desarrolladores pueden encontrar todas las herramientas para crear de forma completa una aplicación independiente.

La ventaja principal del enfoque de cliente ligero es la facilidad de implementación y la posibilidad de trabajar de forma independiente por capas.

Generalmente éstas están estructuradas siguiendo un modelo basado en tres niveles.

- El primer nivel es la interfaz o navegador web. Las funciones que se realizan en este nivel se limitan a enviar las peticiones/consultas que genera el usuario, obtener los datos generados por la aplicación web (servidor) y finalmente representar estos datos en el navegador para el usuario.
- En el segundo nivel se ubicaría la lógica. En este nivel se encuentra en el núcleo principal de la aplicación y es el encargado de dotar a la aplicación web del contenido dinámico, interpretando las consultas generadas por el cliente en el nivel de la interfaz, ejecutando la lógica necesaria para las consultas que se enviarán a la base de datos. Una vez que haya recibido los datos solicitados, los enviará al nivel de la interfaz y los mostrará por medio del navegador.
- En el tercer nivel es donde están los datos. Este nivel podría estar formado por una base de datos, por archivos XML o por cualquier otro tipo de sistema de almacenamiento de datos dependiendo de las necesidades de la aplicación. Sólo se relacionará con el nivel lógico del que recibirá sus peticiones y al que entregará sus respuestas.

Algunos de los problemas más habituales con los que se pueden encontrar los clientes ligeros son:

- Efectos como drag & drop (arrastrar y soltar) y cambio de tamaño de elementos son imposibles de llevar a cabo en un ordenador cliente en un navegador.
- Los navegadores interpretan los lenguajes basados en scripts , como JavaScript, de diferentes maneras, a veces incompatibles entre sí.
- La serialización del estado de la aplicación sólo se consigue utilizando cookies, las cuales no soportan objetos.

Plataformas rich cliente

Los clientes enriquecidos utilizan muchas de las características y facilidades del sistema operativo en el que se ejecuta, lo que soluciona muchas de las carencias de los lenguajes de marcas.

Otra de las características de los clientes enriquecidos es la posibilidad de utilizar una base de datos local, trasladando parte de la carga de procesamiento en el dispositivo, disminuyendo el número de peticiones al servidor y las transferencias por la red.

En este caso se ha pasado de disponer de tres niveles a disponer de cuatro:

- Primero tenemos el navegador web, que contiene el conector o plug-in (rich client) necesario para interpretar los datos que llegan de la aplicación.
- En segundo lugar, tenemos el controlador de la aplicación y la pasarela (gateway). El controlador de aplicación sería la parte de la aplicación que se encarga de interactuar con el cliente y viceversa.
- El siguiente elemento sería el servidor en el que se aloja la aplicación. Este servidor de aplicaciones compilará la aplicación web en caso necesario y permitirá la comunicación de nuestra aplicación con otras aplicaciones o servlets.
- Por último tenemos el almacenamiento de los datos, que habitualmente será una base de datos en la que se almacenarán los datos relacionados con el servicio que ofrece la aplicación.

Estas dos formas diferentes de trabajar comportan que existan componentes diferentes para cada una, en función también de los lenguajes de programación utilizados.

1.1.7. Bibliotecas de componentes para thin y rich clientes

Existen muchas páginas web que ofrecen componentes gratuitos para determinados lenguajes de programación, así como grupos de desarrolladores o comunidades que unifican sus esfuerzos. Éstos pueden ir destinados a programadores individuales o pequeñas empresas que programan con herramientas libres.

1.1.8. Utilidad de los componentes en el desarrollo de aplicaciones

Desarrollo orientado a componentes, este tipo de modelo consiste en desarrollar aplicaciones con interfaces gráficas de usuario a partir del uso de componentes de software,

gratuitos o comerciales, pero que ya existen. Este nuevo modelo de desarrollo busca la limitación de la codificación del nuevo código de programación al mínimo indispensable.

Los beneficios de este tipo de programación son muchos:

- Venta de componentes por empresas especializadas (nuevo campo de negocio).
- Desarrollo más sencillo (en cuanto a tiempo dedicado y, en consecuencia, en cuanto a costes).
- Tasa menos elevada de errores si los componentes que se utilizan pasan por un control de calidad exhaustivo antes de comercializarlos.

1.2. Creación de un componente visual

¿Por qué utilizar componentes visuales?

Si se puede desarrollar una interfaz gráfica de usuario sin escribir ninguna línea de código, se justifica de forma muy convincente la utilización de componentes visuales, siempre que haya quienes desarrollarán las necesidades del programador.

¿Por qué desarrollar nuevos componentes visuales personalizados?

Un componente visual, una vez empaquetado, se podrá utilizar en proyectos futuros, se podrá compartir con otros desarrolladores del mismo proyecto que necesiten herramientas similares o se podrá compartir de forma altruista con el resto de desarrolladores de todo el mundo.

¿Es necesario desarrollar componentes visuales aunque sólo sean para una aplicación?

La respuesta a esta pregunta es un sí , siguiendo la misma línea argumental que en la cuestión anterior. Si en el futuro se quiere modificar la aplicación desarrollada, y los cambios afectan a la parte de código empaquetada en un componente, sólo será necesario distribuir esta biblioteca y no toda la aplicación.

1.2.1. Herramientas para diseñar y crear componentes visuales

Prácticamente la mayoría de los entornos de desarrollo integrados permiten el desarrollo de nuevos componentes visuales, conectores, extensiones, plug-ins o complementos.

Una vez creado un componente visual, será necesario empaquetarlo. Esta acción genera un archivo de tipo dll o msi que se podrá importar en el mismo entorno de desarrollo a otras máquinas o, incluso, a otros entornos de desarrollo.

La mayoría de IDE permiten crear componentes visuales y, algunos menos, empaquetarlos. A continuación, se enumeran algunos de estos IDE:

- Visual Studio .NET.
- MonoDevelop.
- Eclipse, IDE de código abierto multiplataforma.

- NetBeans.
- GLADE

1.2.2. Definición de las propiedades y métodos

Una propiedad es una mezcla entre el concepto de campo de una base de datos y el concepto de método de una clase. Externamente se accede a ella como si se tratara de un campo normal, pero internamente es posible asociar código a ejecutar en cada asignación o lectura de su valor.

Las propiedades son los elementos del componente visual que configuran su aspecto y controlan su comportamiento.

Con las funciones get y set de la programación orientada a objetos se podrá acceder al valor o modificar su valor.

Los métodos son funciones o procedimientos asociados al componente que pueden llamar para que éste realice diferentes acciones de las que obtenga una respuesta.

Los procedimientos y funciones son muy similares. La diferencia es que los procedimientos no devuelven ningún valor, y las funciones siempre devuelven un único valor.

Muchos componentes visuales poseen propiedades en común. Por ejemplo, todos los componentes visuales tienen las propiedades top y left que controlan la posición del componente en el formulario, tanto en tiempo de diseño como en tiempo de ejecución.

- Category . Especifica la forma de clasificar las propiedades en un examinador de propiedades del diseñador visual.
- DefaultValue . Especifica un valor predeterminado para una propiedad.
- Description . Especifica una breve descripción de la propiedad.

A lo largo de este apartado se está haciendo referencia a dos conceptos que, en ocasiones, se tiende a confundir. En ocasiones no queda claro el concepto de propiedad y el de atributo. Y son conceptos y términos distintos:

- Los atributos:
    - Nunca son públicos.
    - Son accesibles únicamente a nivel de programación.
    - Describen el estado interno y el comportamiento de un componente visual en tiempo de ejecución.

- Las propiedades:
    - Son características públicas de sus componentes.
    - Son accesibles a nivel de programación y de diseño de las interfaces gráficas.
    - Describen el estado interno y el comportamiento de un componente visual en tiempo de diseño.

1.2.3. Eventos a los que debe responder un componente

Un evento es una acción determinada que se llevará a cabo vinculada con un componente y que activará cierta funcionalidad. Un ejemplo es hacer un clic sobre un componente.

Esta definición del prototipo de función (o método) al que se desea acceder se realiza por medio de un delegado.

Un evento es un mensaje (o notificación) que lanza un objeto de una clase determinada cuando algo ha ocurrido.

El ejemplo habitual es cuando el usuario hace clic con un dispositivo externo (normalmente un ratón) en un botón de un formulario. Esta acción produce, entre otros, el evento Click del botón pulsado. De esta forma, el botón notifica que esta acción ha pasado y queda a criterio del programador activar una funcionalidad vinculada a este evento.

Los eventos se pueden definir en cualquier clase, y no sólo en clases que mantengan algún tipo de interacción con los usuarios. Eso sí, lo habitual es que se utilicen los eventos con clases que forman parte de la interfaz gráfica que interactúa con el usuario de una aplicación.

1.2.4. Pruebas unitarias y documentación

Las pruebas unitarias también se conocen como pruebas de componentes . Son pruebas sobre partes del código de programación de las aplicaciones informáticas con el objetivo de validar su correcto funcionamiento.

La documentación asociada a un componente será muy similar a la documentación que se puede crear en el desarrollo de una aplicación.

Existen muchos tipos de documentación para completar el desarrollo de una aplicación informática, desde la documentación del código programado, pasando por la documentación técnica hasta la documentación de usuario que indica cómo se explota la aplicación desarrollada.

En el caso de una aplicación informática se puede llegar a encontrar documentación referente a lo siguiente:

- el análisis y los requisitos de la aplicación
- el diseño y su arquitectura
- los temas técnicos
- los usuarios
- el marketing

1.3. Persistencia y serialización

El significado genérico del término persistencia es ' trascendencia en el tiempo y/o en el espacio'.

En el caso de la ejecución de una aplicación informática, se hablaría de guardar el estado de los datos o variables una vez finalizada la ejecución del software.

1.3.1. Persistencia de los componentes visuales

La persistencia de los componentes visuales se define como la capacidad de almacenar los cambios de las propiedades de un componente, esto es, permite que un objeto exista más allá de la ejecución de un programa.

Para persistir debe serializar la información, y una vez serializada se podrá enviar por red, mantener en memoria, enviarla a una impresora, etc.

1.3.2. Serialización

La serialización es la clave para implementar la persistencia. Representa el proceso de almacenamiento del estado de un objeto en un medio. Para ello, se recopila toda la información necesaria del objeto, para convertirse en una secuencia o flujo de datos que, más tarde, permita su recuperación.

Podemos diferenciar distintos tipos de serialización, por ejemplo:

- Binaria
- XML (extensible markup language)

1.3.3 La serialización binaria

La serialización binaria utiliza la codificación binaria para generar una serialización compacta para usos como podría ser el almacenamiento o las secuencias de red basadas en sockets.

La serialización binaria se caracteriza por convertir en un flujo de bytes tanto los miembros públicos como privados junto con el nombre de la clase, incluyendo el código ensamblador, y después se almacena en un flujo de datos. Cuando el objeto es deserializado, se obtiene un clon del objeto original.

1.3.4. La serialización en XML

XML

Lenguaje de marcas extensible para la representación de datos.

La serialización XML serializa las propiedades y campos públicos de un objeto.

Ésta es una de las diferencias con la serialización binaria que, tal y como se ha visto, guarda tanto las propiedades públicas como las privadas.

1.4. Empaquetado de componentes

El empaquetado de componentes abrió muchas vías, facilitó en gran medida el proceso de compartición de componentes, facilitó el proceso, distribución y hecho lo más transparente posible para los usuarios finales. Además, el proceso de empaquetar un componente es un proceso automático que facilitará las actualizaciones y mantenimiento de las aplicaciones.

Antes del proceso de creación y empaque de un componente, habrá que tener en cuenta algunos elementos como pueden ser:

- Empaquetado de fuentes, objetos y ejecutables.
- Arquitectura de destino del paquete.
- Información relativa al paquete (técnica, usuario, etc.)
- Distribución en el destino del paquete.
- Firma digital.

1.4.1. Empaquetado de un thin cliente

jQuery es una biblioteca de componentes y funcionalidades implementada en JavaScript, pensada para interactuar con los elementos de una web a través del DOM. Lo que la hace tan especial es la sencillez y tamaño reducido.

Para poder utilizar la biblioteca jQuery deberemos hacer referencia a los archivos de JavaScript que disponen del código fuente de los componentes.

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.4/jquery.min.js"> </script> <script src="http://ajax.googleapis.com/ajax/libs/jqueryui/1.8/jquery-ui.min.js"> </script>

1.4.2. Empaquetado de un rich cliente

Empaquetar un componente es una de las mejores y más recomendadas maneras de distribuir los componentes a usuarios de cierto entorno, o ponerlos a disposición en un espejo para que se puedan usar en el desarrollo de aplicaciones.

Una vez creado un componente en una plataforma rich client, para poder distribuirla será necesario crear un instalador o una biblioteca para que este componente pueda ser importante o pueda ser instalado en otros entornos u otras máquinas. Las herramientas que permiten esta función suelen incorporar funcionalidades que ayudan a empaquetar el componente, con wizards o auxiliares que guían a los desarrolladores en el empaquetado.

Dos de los tipos más habituales son el de creación de un paquete de instalación y el de creación de dlls:

- Paquete de instalación estándar. Es un tipo de empaquetado que creará un archivo .exe o un archivo setup.exe o un archivo .msi que permitirá distribuir el componente en otros entornos que utilicen la misma IDE para el desarrollo de aplicaciones.

- Archivo o biblioteca. Se generará un tipo de archivo dll o biblioteca dependiendo del lenguaje de programación y el entorno de desarrollo.