DI01.- Confección de interfaces de usuario.

1.- Elaboración de interfaces de usuario.

La interfaz de usuario es el elemento de una aplicación informática que permite al usuario comunicarse con ella.

El desarrollo de aplicaciones hoy día no puede ser entendido sin dedicar un porcentaje bastante importante de tiempo a planificar, analizar, diseñar, implementar y probar sus interfaces de usuario ya que son el medio fundamental por el cual el usuario puede comunicarse con la aplicación y realizar las operaciones para las que ha sido diseñada. Existen diferentes tipos de interfaces de usuario:

* **Textuales**. La comunicación se produce por medio de la inserción de órdenes escritas en un intérprete de órdenes.
* **Gráficas**. La interfaz consta de un conjunto de elementos visuales, como iconos o menús con los que se interacciona, normalmente, mediante un elemento apuntador (el ratón, por ejemplo). Son las más habituales y tienen a gala haber popularizado el mundo de la informática para usuarios noveles.
* **Táctiles**. La comunicación se produce mediante a través de un dispositivo táctil, generalmente una pantalla que puede reaccionar ante la presión táctil de elementos apuntadores o incluso de los dedos. Se usan habitualmente en dispositivos móviles, terminales de puntos de venta y para el diseño de gráficos por ordenador.

Una interfaz gráfica está formada por un conjunto de ventanas, llamadas **formularios**, y que dentro de ellos podemos colocar diferentes elementos visuales, que se denominan **controles o componentes** con los que al interactuar damos órdenes o podemos recibir información.

2.- Componentes.

Una interfaz gráfica se comporta como un todo para proporcionar un servicio al usuario permitiendo que éste realice peticiones, y mostrando el resultado de las acciones realizadas por la aplicación. Sin embargo, se compone de una serie de elementos gráficos atómicos que tienen sus propias características y funciones y que se combinan para formar la interfaz. A estos elementos se les llama **componentes** o **controles**.

Algunos de los componentes más típicos son:

* **Etiquetas**: Permiten situar un texto en la interfaz. No son interactivos y puede utilizarse para escribir texto en varias líneas.
* **Campos de texto**: cuadros de una sola línea en los que podemos escribir algún dato.
* **Áreas de texto**: cuadros de varias líneas en los que podemos escribir párrafos.
* **Botones**: áreas rectangulares que se pueden pulsar para llevar a cabo alguna acción.
* **Botones de radio**: botones circulares que se presentan agrupados para realizar una selección de un único elemento entre ellos. Se usan para preguntar un dato dentro de un conjunto. El botón marcado se representa mediante un círculo.
* **Cuadros de verificación**: botones en forma de rectángulo. Se usan para marcar una opción. Cuando está marcada aparece con un tic dentro de ella. Se pueden seleccionar varios en un conjunto.
* **Imágenes**: se usan para añadir información gráfica a la interfaz.
* **Password:** es un cuadro de texto en el que los caracteres aparecen ocultos. Se usa para escribir contraseñas que no deben ser vistas por otros usuarios.
* **Listas**: conjunto de datos que se presentan en un cuadro entre los que es posible elegir uno o varios.
* **Listas desplegables**: combinación de cuadro de texto y lista, permites escribir un dato o seleccionarlo de la lista que aparece oculta y puede ser desplegada.

2.1.- Bibliotecas de componentes.

Los componentes que pueden formar parte de una interfaz gráfica se suelen presentar agrupados en bibliotecas con la posibilidad de que el usuario pueda generar sus propios componentes y añadirlos o crear sus propias bibliotecas. Se componen de un conjunto de clases que se pueden incluir en proyectos software para crear interfaces gráficas. El uso de unas bibliotecas u otras depende de varios factores, uno de los más importantes, por supuesto, es el lenguaje de programación o el entorno de desarrollo que se vaya a usar. Dependiendo de esto podemos destacar:

**JAVA Fundation Classes (JFC):**

Las JFC incluyen las bibliotecas para crear las interfaces gráficas de las aplicaciones Java y applets de Java.

* **AWT**: Primera biblioteca de Java para la creación de interfaces gráficas. Es común a todas las plataformas, pero cada una tiene sus propios componentes, escritos en código nativo para ellos. Prácticamente en desuso.
* **Swing**: Surgida con posterioridad, sus componentes son totalmente multiplataforma porque no tienen nada de código nativo, tienen su precursor en AWT, de hecho, muchos componentes swing derivan de AWT, basta con añadir una J al principio del nombre AWT para tener el nombre swing, por ejemplo, el elemento **Button** de AWT tiene su correspondencia swing en **Jbutton** aunque se han añadido gran cantidad de componentes nuevos. Es el estándar actual para el desarrollo de interfaces gráficas en Java.

Además, existen bibliotecas para desarrollo gratifico en 2D y 3D y para realizar tareas de arrastrar y soltar (drag and drop).

**Bibliotecas MSDN de Microsoft (C#, ASP, ...)**:

**.NET framework**: hace alusión tanto al componente integral que permite la compilación y ejecución de aplicaciones y webs como a la propia biblioteca de componentes que permite su creación. Para el desarrollo de interfaces gráficas la biblioteca incluye ADO.NET, ASP.NET, formularios Windows Forms y la WPF (Windows Presentation Fundation).

**Bibliotecas basadas en XML**:

También existen bibliotecas implementadas en lenguajes intermedios basados en tecnologías XML. Normalmente disponen de mecanismos para elaborar las interfaces y traducirlas a diferentes lenguajes de programación, para después ser integradas en la aplicación final.

**Otras API (Application  Programming Interface, Interfaz de programación)**

También hay que destacar que existen otras bibliotecas o API  como son:

* **DirectX**: plataforma Microsoft creada para facilitar el manejo de los elementos multimedia. Consta a su vez de otras API como son Direct3D, Direct Graphics, Direct sound, Direct Input, DirectPlay, DirectShow, DirectMusic, DirectSetuo y DirectCompute.
* **GTK** (GIMP tOOL KIT): biblioteca del equipo GTK+. El entorno gráfico de GNOME utiliza esta librería. Maneja widgets como ventanas, etiquetes, pestañas, etc y se puede utilizar en lenguajes C, C++, C#, Java, Python, Ruby.
* **QT**:es utiliza por el entorno gráfico KDE. Utiliza lenguaje de programación C++ y puede ser integrado en otros lenguajes. Se utiliza porque también se utilizan en sistemas empotrados como automoción, aereonavegación y aparatos domésticos.

3.- Herramientas para la elaboración de interfaces.

Las bibliotecas son esenciales para desarrollar aplicaciones y suelen integrarse en entornos de desarrollo de software que facilite el trabajo con algún sistema de ventanas, en el que se incluyen diferentes regiones rectangulares, llamadas "ventanas" cuya parte central es un conjunto de herramientas (toolkit). A este tipo de herramientas se le llama IDE y facilitan la inclusión de componentes gráficos de bibliotecas en aplicaciones de manera intuitiva y sirven como interfaz de usuario.

A continuación, se listan varios de los IDE más utilizados en la actualidad:

* **Microsoft Visual Studio**: Es un entorno para el desarrollo de aplicaciones en entornos de escritorio, web y dispositivos móviles con la biblioteca .NET framework de Microsoft. Permite el uso de diferentes lenguajes de programación como C++, C# o ASP. En la actualidad se pueden crear aplicaciones para Windows 7, 8 y10, aplicaciones web y RIA (Rich Internet Aplications).
* **NetBeans**: IDE distribuido por Oracle bajo licencia GNU GPL. Está desarrollado en Java, aunque permite crear aplicaciones en diferentes lenguajes, Java, C++, PHP, Ajax, Python y otras.
* **Eclipse**: IDE creado inicialmente por IBM ahora es desarrollado por la fundación Eclipse. Se distribuye bajo la Licencia Pública Eclipse (EPL). Tiene como característica más destacable su ligereza ya que se basa en módulos, partiendo de una base muy sencilla con un editor de texto con resaltado de sintaxis, compilador, un módulo de pruebas unitarias con JUnit, control de versiones con CVS, integración con Ant, asistentes (wizards) para creación de proyectos, clases, tests, etc. y refactorización. Si se precisan funcionalidades más allá, éstas se incluirán como módulos aparte que van completando el IDE.
* **JDeveloper**: Desarrollado por Oracle para los lenguajes Java, HTML, XML, SQL, PL/SQL, Javascript, PHP, Oracle ADF, UML y otros. Desde 2001 está basado en Java. Está cada vez en más desuso, ya que NetBeans ofrece mayores opciones.
* **Aptana Studio**: basado en eclipse incluye un importante motor para el desarrollo de interfaces web.
* **Oracle Database**: permite crear interfaces para acceder a bases de datos.
* **Dreamweave**r: no solo permite el diseño de interfaces web, sino que también permite el desarrollo de aplicaciones basadas en ASP.NET, PHP, Javascript, CSS, ColdFusion, etc.
* **Komodo Edit**: sus características dependen de un intérprete de phyton.

En esta unidad vamos a utilizar NetBeans, junto con la biblioteca swing para la generación de interfaces. Se ha seleccionado este entorno por varios motivos, en primer lugar, permite crear aplicaciones tanto de escritorio, como web como para dispositivos móviles y además se distribuye bajo licencia CDDL y GPL2. Es multiplataforma e incluye varios lenguajes de programación.

Una de sus principales ventajas es que resuelve por sí mismo el tema de la colocación de los componentes en una interfaz gráfica, aspecto de cierta complejidad a la hora de programar, de forma que el desarrollador sólo tiene que colocar los controles usando el ratón y el *IDE* se encarga de programarlo.

También permite la inclusión de componentes creados por otros desarrolladores que completan la paleta swing/AWT.

4.- Contenedores.

Un **formulario** es una ventana que dispone de tres botones para minimizarse, maximizarse o cerrarse, una barra de título y está delimitado por unos bordes. Es la base para crear una aplicación de escritorio. Sobre él se añadirán controles o componentes, sencillos como botones o cajas de texto o más complejos, como barras de menú o rejillas de datos, que le dan funcionalidad.

4.1.- Contenedores de primer nivel.

Una aplicación de escritorio se compone de una serie de formularios. Para crear un formulario, tendrás que usar un **contenedor** Java que es un componente que permite incluir otros componentes, incluidos otros contenedores, que se usarán para distribuir los controles. Los contenedores forman una **estructura jerárquica**.

Un formulario está formado por un contenedor de nivel superior. Este tipo incluye un panel de contenido (contentpane) que permite añadir otros componentes, incluidos otros contenedores que se utilicen facilitar la distribución de elementos.

Como contenedor de nivel superior de un formulario puedes elegir entre una ventana (JFrame), un diálogo (JDialog) o un applet (JApplet), según la necesidad. Todos estos componentes derivan, en la jerarquía de clases de java, de Window que representa una ventana típica.

* **Ventana (JFrame)**: es un formulario con título, los botones para maximizar, minimizar o cerrar y borde. Aparece un icono por defecto en forma de taza de café que puedes modificar, y puede contener una barra de menú.
* **Dialogo (JDialog):** formularios que se suelen usar para solicitar información al usuario. Su principal característica es que pueden ser modales o no, una ventana modal recibe todas las entradas de usuario e impide que se active otra ventana.
* **Applet (JApplet):** ventana que ejecuta una aplicación Java, en contexto.

En una aplicación de escritorio se suele crear una ventana principal de tipo JFrame y si necesitamos ventanas secundarias utilizaremos otras ventanas o diálogos.

[Ejemplos y ejercicio](DI01_CONT_R16_contenedores.1.pdf)

Si quieres saber un poco más acerca de los contenedores de primer nivel [sigue este enlace](https://download.oracle.com/javase/tutorial/uiswing/components/toplevel.html)

4.2.- Contenedores secundarios.

También se interpretan como diálogos los siguientes componentes:

* **Panel de opciones (JOptionPane)**: genera ventanas con botones para responder cuestiones con respuestas del tipo si-no, aceptar-cancelar, aceptar, etc.
* **Selector de archivos (JFileChooser)**: permite seleccionar un archivo del sistema de archivos del equipo donde se ejecuta la aplicación.
* **Selector de colores (JColorChooser)**: permite seleccionar entre un conjunto de colores y devolverlo usando el código adecuado.

Puedes usar otro tipo de contenedores para distribuir el resto de los controles que se incluyen en la ventana principales, entre los más habituales tienes:

* **Paneles (JPanel)**: representa un contenedor intermedio, cuya función principal es la de colocar controles.
* **Barra de menús (JMenu)**: permite la creación de menús complejos de opciones.
* **Barra de herramientas (JToolBar)**: se utiliza para contener iconos de acceso a las opciones de la aplicación.
* **Pestañas (JTabbedPane)**: tipo particular de panel que permite la distribución de elementos en pestañas o tarjetas.
* **Paneles deslizables (JScrollPane)**: tipo especial de panel que permite desplazar sus contenidos de manera automática.
* **Ventanas internas (JInternalFrame)**: ventanas hijas que no pueden rebasar los límites de la ventana padre donde se han creado. Se usan en aplicaciones que tienes varios documentos abiertos simultáneamente.
* **Paneles divididos (JSplitPane)**: permite visualizar dos componentes, uno a cada lado, asignando espacio dinámicamente a cada uno.

5.- Componentes de la interfaz.

Los componentes o controles gráficos de un formulario, **son elementos gráficos que se anidan en los contenedores** para formar aplicaciones. **Se utiliza para mostrar información**, como etiquetas o imágenes, listas (**componentes pasivos**) o árboles, pero, sobre todo, **para recabar información del usuario**, como cuadros de texto, botones, o listas de selección (**componentes activos**).

Un componente **se reconoce por su clase**, que define su aspecto y funcionalidad y por su **nombre,** que lo identifica dentro de la aplicación. Puesto que es un objeto, según la clase a la que pertenezca **tendrá una serie de propiedades que podremos modificar** para adaptar el componente, por ejemplo, el texto mostrado, o el color.

La **colocación de componentes** en el formulario **se rige por** unas reglas, denominadas **Layouts**, que **establecen el orden y la posición en la que deben ser mostrados**, pudiendo ser en torno a **los límites del formulario** (norte, sur, este y oeste), en forma de **rejilla**, o **fluidos**, **uno tras otro, en una o varias filas**.

Cuando un **componente** es **activo,** es susceptible de interactuar con el usuario, **se gestiona mediante** lo que se conoce como **manejo de eventos**, **una acción** sobre el componente que **debe provocar** una respuesta se conoce como **evento**, **la gestión** de la respuesta **se realiza a través de los manejadores de eventos**, **que son** unas **funciones específicas que se asocian a un** elemento del componente denominado **escuchador**, en el que se programa la acción a realizar.

[Documentación de Java Swing](https://docs.oracle.com/javase/8/docs/api/javax/swing/package-summary.html)

5.1.- Añadir y eliminar componentes de la interfaz.

Los componentes se pueden añadir desde la paleta, que, si recordamos suele anclarse a la derecha de la interfaz del IDE NetBeans.

A la derecha tienes la lista de controles para añadir a una interfaz en NetBeans. Están organizados en las siguientes categorías:

* **Contenedores swing**: son secundarios en la jerarquía de contenedores y se usan para distribuir y organizar el resto de controles.
* **Controles Swing**: básicos para crear una interfaz útil para comunicarse con el usuario y mostrar o solicitar información.
* **Menús Swing**: incluyen los controles necesarios para crear menús de aplicación complejos, con varios bloques, elementos activos e inactivos, etc. y menús contextuales (Popup Menu)
* **Ventanas Swing**: permiten añadir a la aplicación ventanas (JFrame), diálogos (JDialog), selectores de ficheros y colores (JFileChooser y JColorChooser) y paneles de opciones (JOptionPane) para crear diálogos que se contestan con Sí/No.

Para añadir componentes, seleccionaremos el componente en la paleta y lo arrastramos sobre la interfaz. El control aparece en la interfaz con su aspecto por defecto que puedes modificar. Se puede recolocar y cambiar su tamaño una vez este en la interfaz

Al colocar un control sobre un formulario aparecen unas guías que te permiten colocarlo con más facilidad, esto será así mientras que tengas activa la opción **diseño libre**, lo puedes comprobar en el inspector haciendo clic con el botón secundario en el nodo raíz de la interfaz y seleccionando *activar gestor de distribución*. Si **mueves** un control estas guías te permitirán relacionarlo con otros componentes para que lo puedas alinear mejor.

Conforme vas colocando controles en el panel del formulario éstos aparecen reflejados, además, en el **Inspector**, de forma que los seleccionas tanto haciendo clic sobre ellos como sobre su nombre. Trabajar con el inspector facilita colocar controles dentro de contenedores porque admite operaciones de arrastrar y soltar.

Para **eliminar** un control basta con seleccionarlo y pulsar Suprimir, o bien seleccionar la opción Suprimir del menú contextual (botón derecho del ratón).

5.2.- Modificación de propiedades.

A un control que se encuentra en el formulario se le puede modificar sus propiedades para adaptarla. Con el control seleccionado accedes a sus propiedades en el panel propiedades, que lógicamente, será diferente para cada tipo de control.

Se suele comenzar por modificar el **nombre** del control para localizarlo con más facilidad en el código de la clase que genera el formulario. Para hacerlo puedes sacar el menú contextual del control en el Inspector y seleccionar *cambiar nombre de la variable...*

Otra propiedad muy común es el **ToolTipText**, en el que aparece una breve descripción de para qué sirve el control. Lo puedes establecer en la propiedad **ToolTipText**.

5.3.- Añadir funcionalidad desde NetBeans.

Creamos interfaces para permitir que el usuario pueda interactuar con la aplicación, pero siempre será necesario añadir código para darles la funcionalidad para la que han sido creadas. Cuando usamos un entorno integrado como NetBeans parte de este código se genera de forma automática facilitando en gran medida el trabajo del desarrollador, pero tendrás que abrir este código y modificar algunas cosas para que el formulario realice las tareas para las que ha sido diseñado.

5.4.- Ubicación y alineamiento de los componentes.

Internamente la herramienta emplea el mecanismo de Java para disponer los elementos llamado Layout, distribución o diseño. Swing dispone de ocho tipos de distribuciones:

* **BoderLayout**: aloja los componentes en los límites del formulario, por lo que cuando los colocamos debemos indicar si van al norte, sur, este u oeste.
* **GridLayout**: Diseña mediante una rejilla, en la que los componentes se organizan por filas y columnas.
* **GridBagLayout**: semejante a **GridLayout**, pero permite a un componente que ocupe más de una celda.
* **CardLayout**: diseño por paneles. Permite la colocación de distintos componentes en momentos distintos de la ejecución.
* **BoxLayout**: diseño en caja. Coloca los componentes en una fila o columna ajustándose al espacio que haya.
* **FlowLayout**: diseña alojando los componentes de izquierda a derecha mientras quede espacio, si no queda pasa a la fila siguiente.
* **GroupLayout**: se creó para ser utilizado en herramientas de diseño gráfico de interfaces. Trabaja por separado la distribución vertical y horizontal para definir exactamente el posicionamiento de los componentes. Se utiliza en NetBeans.
* **SpringLayout**: es muy flexible y se usa también para herramientas de diseño gráfico de interfaces. En este caso se especifican las relaciones entre los límites de los componentes bajo su control.

Programar el diseño de un formulario es una de las tareas más arduas en Java, si bien está ampliamente superado gracias al uso de IDEs que facilitan la colocación de componentes a golpe de ratón y sin necesidad de escribir código. Por ejemplo, NetBeans, usa el diseño **GroupLayout**.

[Guía visual de los gestores de diseño.](https://download.oracle.com/javase/tutorial/uiswing/layout/visual.html)

En el siguiente enlace puedes encontrar los archivos del proyecto de NetBeans con la aplicación que hemos estado haciendo. Te recomiendo que le eches un vistazo para comprobar los nombres que se les han puesto a las variables, los ToolTipText, etc.

[Enlace al ejemplo](https://www.adistanciafparagon.es/pluginfile.php/35777/mod_resource/content/0/AplicacionEscritorio.zip)

5.5.- Enlace de componentes a orígenes de datos.

Uno de los aspectos que aporta mayor flexibilidad y potencia a una aplicación es que pueda acceder a la información contenida en una base de datos.

Los pasos a dar para conseguir que una aplicación java, utilizando NetBeans, incorpore un formulario basándose en los datos almacenados en una tabla de una base de datos son:

1. Tener instalado MySQL y que se encuentra en ejecución. A continuación, crearemos la base de datos. La base de datos que vamos a utilizar se llamará Agenda. Puedes importar la estructura de la base de datos junto con sus datos utilizando el fichero que encontrarás en apartado Recursos que aparece al final de la página ([BD\_agenda.sql](BDAgenda_sql.zip))
2. Recomendación: Antes de comenzar a crear el proyecto en NetBeans, accede al apartado Services o Prestaciones y comprueba que podemos ver que aparece Servidor MySQL y que podemos acceder a la base de datos Agenda. En el caso de que no aparezca debes de registrar servidor MySQL e iniciarlo (para ello accedemos al menú contextual de la opción Base de datos y realizamos las dos acciones.)
3. Crearemos un nuevo proyecto NetBeans de tipo *Java Application* que se llame Agenda. No crearemos clase principal al crearlo. Añadimos un paquete para almacenar todas las clases que se van a generar de forma automática. Se va a llamar *sources* y lo vamos a crear dentro de Paquete de fuentes. Añadimos un Formulario de ejemplo/detalle que podemos encontrarnos dentro de la categoría *Formularios de interfaz gráfica de Swing*.  Le asignaremos el nombre Contactos.
4. A continuación, nos pedirá el acceso a la base de datos. En la conexión con la base de datos marcamos la cadena de conexión jdbc:mysl...... que nos aparecerá en el desplegable. Si no nos apareciera nuestra conexión con MySQL, este desplegable nos permite crear una conexión nueva. En el apartado Tabla de base de datos, seleccionamos la tabla Contactos y en el apartado Campos a incluir nos aseguramos de tener incluido las columnas que queremos utilizar en el formulario.
5. En el apartado *Crear área de detalle como* seleccionaremos la opción cuadro de texto.
6. Finaliza el asistente pulsando Terminar, en el formulario Agenda aparecerán una tabla para mostrar el resultado de la consulta.

Si seleccionas la tabla que ha aparecido en el formulario y observas el Inspector verás que se llama **masterTable** (Jtable). Esta tabla en concreto tiene tres columnas y se le asigna contenido a partir de un elemento, que también puedes ver más abajo en el Inspector llamado **list** (JList) que se rellena a partir de una consulta llamada Query (Jquery), en cuyas propiedades aparece la consulta ejecutada que es SELECT c FROM contactos c. Haz clic con el botón secundario sobre la tabla >> Enlazar >> Elements, verás cómo enlaza con la lista y como se obtienen de ella los tres campos de la tabla.

Por otra parte, se han añadido tres etiquetas y tres campos de texto al formulario, los campos de texto enlazan (botón secundario >> Enlazar >> Text) con la tabla maestra en el enlazado de fuente de donde se saca el valor del campo del registro actual mediante la expresión **${SelectedElement.nombre\_campo}** de esta manera cada vez que se cambia el registro activo de la tabla se modifica el contenido de los campos de texto.