**Patrones de arquitectura**

Es la descripción de los elementos y el tipo de relación que tienen junto con una serie de restricciones o normas sobre cómo se deben usar.

En otras palabras, es un esquema de organización esencial para un sistema de software, compuesto por otros sistemas, responsabilidades e interrelaciones.

A diferencia de los patrones de diseño, estos tienen un nivel de abstracción mayor.

Los patrones de arquitectura de aplicaciones gráficas más usados son

* Programación por capas
* Tres niveles
* Arquitectura de microservicios
* Arquitectura de microkernel
* Invocación implícita
* Arquitectura en pizarra
* Arquitectura dirigida por eventos, Presentación-abstracción-control
* Peer-to-peer
* Arquitectura orientada a servicios
* Objetos desnudos
* Modelo Vista Controlador

El **modelo vista controlador (MVC)**: Es aquel patrón que se suele utilizar para implementar interfaces de usuario, datos y lógica de control. Se basa en la separación entre la lógica de negocios y su visualización, de forma que proporciona una mejor división del trabajo y una mejora del mantenimiento de la aplicación.

El MVC se compone de tres secciones:

* **Modelo**: Maneja datos y lógica de negocios.
* **Vista**: Se encarga del diseño y presentación.
* **Controlador**: Enruta comandos a los modelos y vistas

- El **Modelo** en el MVC es el que define qué datos debe contener la aplicación, de forma que si el estado de esos datos cambia, el modelo notificará a la vista y si es necesario al controlado para que apliquen los cambios en caso de que sean necesarios.

- La **Vista** en el MVC es dónde se define cómo se deben mostrar los datos de la aplicación, para lo que recibirá una serie de datos y elegirá según ciertas condiciones *de diseño*, que deben mostrar y que deben ocultar.

- El **Controlador** en el MVC es el contiene la lógica que actualiza el modelo y/o vista en base a las respuesta y peticiones de los usuarios de la aplicación.El controlador contiene una lógica que actualiza el modelo y/o vista en respuesta a las entradas de los usuarios de la aplicación.

**Lenguajes descriptivos**

Un **lenguaje descriptivo o declarativos** es aquel que se basa en el paradigma de programación declarativa, el cual se caracteriza porque el programador declara y especifica condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describe el problema, modelando así la programación lógica y la programación funcional.

El lenguaje descriptivo describe el inconveniente a solucionar, pero no las instrucciones necesarias para hacerlo.

Existe una gran variedad de lenguajes descriptivos pero los más famosos y los que más se utilizan hoy en día son: HTML, XHTML, XML, CSS, Etc…

**Librerías de componentes nativas y multiplataforma**

Una **librería** es un conjunto de funciones, clases y recursos que se pueden utilizar en el desarrollo de software. Pueden estar creadas por otros desarrolladores o pueden ser creados por los mismos programadores para su propio uso.

Existe una gran variedad de formas de clasificar a las librerías por ejemplo:

* En función de su propósito
* En función de su creación
* En función del lenguaje en el que se ha desarrollado
* En función de su uso
* En función de su dinamismo
* Etc.

Una **librería estática** es aquella que se enlazan al compilar, de forma que se alojan "dentro" del ejecutable final.

Una **librería nativa** es aquella que se ha diseñado específicamente para un dispositivo y sirve exclusivamente para ese sistema y ese lenguaje.

Para entenderlo mejor se puede decir que el play store es una aplicación que contiene librerías nativas de Android ya que no sirve para otros sistemas operativos.

Una **librería multiplataforma** a diferencia de la anterior es aquella que está diseñada para que se pueda utilizar en varios dispositivos independientemente del lenguaje en el que estuviera creado inicialmente.

Por ejemplo las librerías que contiene el juego de móvil Candy Crush son multiplataforma ya que está disponible para Android y para iOS.

**Tipos de herramientas de desarrollo**

Entre otros motivos, el desarrollo de las GUI depende bastante **del lenguaje de programación** que se utilice y de su entorno de desarrollo.

No es lo mismo desarrollar una aplicación informática con un lenguaje de programación de **tercera generación** (se utilizan bibliotecas para tener acceso a un entorno gráfico) o de **cuarta generación** (incluyen componentes gráficos y de acceso a datos incorporados). → Necesitan herramientas de software diferentes.

Otro factor a tener en cuenta a la hora de desarrollar las interfaces es el entorno dónde se va a utilizar la GUI. No es lo mismo desarrollar una aplicación para un entorno Web que para un entorno de ventanas (Winforms). → Los componentes, lenguajes, metodología de trabajo y posibilidades son diferentes en ambos casos.

Para implementar las GUI se utilizan herramientas de creación y edición, las cuales se definen como, un tipo de software que ayudará al programador a desarrollar interfaces que cumplan todas las necesidades indicadas en la fase de diseño.

Cabe resaltar que para implementar una GUI se tienen en cuenta todas las fases de desarrollo de un proyecto informático.

Por último debemos saber que el desarrollo de las GUI debe tener en cuenta muchos aspectos ya que engloba el diseño gráfico, de lingüística, de programación, de tecnología informática etc. → No es un trabajo sencillo de realizar.

Existe una gran variedad de herramientas que nos permite el desarrollo de las GUI. En general las podemos clasificar en función de:

* El **lenguaje de programación** que se utilice
* El **entorno de desarrollo** que se utilice
* Si son **herramientas privativas o libres**.

Las principales ventajas que nos proporcionan las herramientas de desarrollo de GUI son:

* Incorporación de varios elementos preprogramados que facilitan la creación de GUI más complejas. → Se desarrolla de forma sencilla. → Ejemplo: Añadir un menú o un icono se reduce en desplazar ese elemento hasta el panel principal y luego modificar sus propiedades. → Los elementos pueden ser para fomentar la interacción con el usuario (ayudas, acciones sobre cursores, etc) o elementos de configuración de interfaces (iconos, ventanas, menús, etc)
* Gestión de errores por parte del usuario → Procesar excepciones, gestionar y controlar los accesos, gestionar los dispositivos de entrada, etc.
* Herramientas tipo WYSIWYG → Permite visualizar los resultados de forma inmediata.

Algunas de las herramientas de edición de GUI son:

* **Microsoft Visual Studio:** Entorno de desarrollo integrado de *Microsoft* para SO Windows. Privativo. Lenguajes de programación: Visual C++, Visual C#, Visual J#, ASP.NET y Visual Basic .NET.
* **Eclipse:** Desarrollado inicialmente por *IBM* aunque ahora está gestionado por una fundación sin ánimo de lucro → gratuito y open source. Lenguajes de programación: Java, C++, Perl y PHP. Multiplataforma.
* **IDE NetBeans:** Desarrollado inicialmente por Sun. Gratuita y open source. Lenguajes de programación: Java, JSP, C++ y PHP. Multiplataforma.
* **MonoDevelop**: Gratuito y de código abierto. Patrocinado por Novell. Lenguajes de programación: C#, JAva o Boo. Multiplataforma
* **SharpDevelop**: Entorno de desarrollo de código abierto. Lenguajes de programación: C#, Visual Basic .NET y Boo. Para SO Windows.

**Componentes de las interfaces gráficas**

Una **interfaz de usuario** es un conjunto de elementos (que pueden pertenecer al software o al hardware) que ofrecen una información al usuario, y permiten, además, la interacción (física o lógica) entre el usuario y el ordenador, por medio de un dispositivo periférico o un enlace de comunicaciones.

Dentro de las interfaces de usuario diseñadas para aplicaciones informáticas, nosotros nos vamos a centrar en las interfaces de usuario gráficas o **GUI** (graphical user interface).

**Las interfaces gráficas de usuario (GUI)** son aquellas que utilizan elementos gráficos, como pueden ser menús, ventanas o diálogos, además del uso de otros recursos del sistema informático (periféricos como el teclado, el ratón o el sonido ) para permitir al usuario interactuar con el ordenador de forma muy sencilla e intuitiva.

Algunas de las principales funciones que pueden desempeñar las GUI son las siguientes:

* Configuración de las interfaces gráficas de usuario y del entorno de trabajo
* Control de acceso a una aplicación informática
* Sistemas de ayuda interactivos.
* Gestión y manipulación de directorios y archivos de un sistema operativo.
* Arranque y cierre de un sistema informático mediante un sistema operativo.
* Intercambio de datos entre distintas aplicaciones.
* Comunicación entre sistemas informáticos.
* Ayuda al desarrollo de aplicaciones informáticas.
* Ayuda al diseño y desarrollo de interfaces gráficas de usuario.
* Gestión y manipulación de las funcionalidades que se puedan configurar en los sistemas informáticos.

El objetivo principal de una interfaz de usuario consiste en comunicarse de manera sencilla y agradable con el usuario por medio de un dispositivo de entrada/ salida.

A pesar de que no todas las interfaces han sido sencillas ni agradables de utilizar (interfaces de líneas de instrucciones <CLI>), siempre cuentan con ciertas funcionalidades que son mejores para usuarios más expertos.

**Las interfaces de líneas de instrucciones (CLI)**  son aquellas que permiten a los usuarios interactuar con el ordenador por medio de líneas de texto simples que contienen instrucciones.

Se utilizaban sobre todo en los sistemas operativos antiguos cuando no se tenían los conocimientos suficientes para usar las GUI.

Actualmente se siguen utilizando ya que una instrucción compleja es bastante difícil de representar mediante los elementos que ofrecen las interfaces gráficas. Algunos ejemplos que lo incluyen son los routers o los conmutadores.

Cabe destacar que todos los sistemas operativos ofrecen la posibilidad de trabajar mediante un intérprete de instrucciones por ejemplo shell de UNIX o powershell de Windows…

Debemos tener en cuenta que no todas todas las funcionalidades se pueden representar mediante interfaces de usuario. Por ejemplo aquellas que requieran muchas instrucciones serán difícilmente programables mediante interfaces gráficas; sin embargo, con una pequeña programación Batch en Windows o los shell scripts en UNIX, se programarán con facilidad.

Por último debemos resaltar que el diseño de las interfaces de usuario es una **parte crítica**. Es necesario que se cumplan todos los **requisitos especificados** y que sea **sencilla e intuitiva** para el usuario ya que si no se cumplen, difícilmente se conseguirá una interfaz adecuada para ser usada de forma agradable y sencilla por los usuarios.

Las **GUI** están compuestas por **elementos** propios, comunes a otras GUI desarrolladas, y que disponen de ciertas características y propiedades.

A través de esto se facilita el desarrollo de las GUI de las aplicaciones informáticas, ofreciendo la posibilidad de cumplir la mayoría de las características definidas.

Para que el usuario pueda utilizar estos elementos, necesita usar algún dispositivo de entrada/salida (teclado, ratón, touchpad, micrófono, altavoz, pantalla, impresora…).

Para algunos de los elementos que ofrecen las GUI vamos a necesitar ciertos dispositivos.

En conclusión, cada GUI utiliza los elementos en función de su entorno de trabajo y por tanto no todas las GUI tienen los mismo elementos.

Los elementos más comunes que tienen las GUI son los siguientes:

Ventanas, Cuadros de diálogo, Asistentes, Menús Pestañas, Barra de herramientas, Iconos, Entornos de trabajo, Controles y Tipografía

Las **ventanas(windows)** son de los elementos más importantes de las GUI. Son normalmente bidimensionales y rectangulares, y se ubican en el escritorio del SO. Cuando hay más de una ventana abierta la que vemos es la que está activa y el resto están escondidas por debajo (profundidad).

A través de las ventanas podemos trabajar a la vez en distintos entornos que pueden ser de la misma o diferentes aplicaciones. → Excel y navegador

Si se utilizan dos ventanas de una misma aplicación, tenemos más de una interfaz abierta al mismo tiempo, por tanto podemos comparar la ejecución de dos funcionalidades o por ejemplo tener una para consultar datos y otra para modificarlos.

Las principales características que nos ofrecen las ventanas son:

* Ofrecer información al usuario de forma organizada
* Permite navegar por las funcionalidades de una aplicación visualizando la información de forma jerarquizada.
* Permite gestionar y manipular la información de forma bastante sencilla.
* Pueden ser manipuladas como al usuario le interese, maximizándola o minimizándola, moviéndolas o cerrándolas, navegando o haciéndolas de mayor tamaño...

Contienen GUI que incorporan otros elementos propios como menús, pestañas, barras de herramientas, objetos gráficos o multimedia, que permitirán tanto la salida como la entrada de datos

Los elementos básicos por los que están compuestos las ventanas son:

* **Marco**: Sirve para distinguir lo que pertenece a la ventana y lo que no y ayuda a redimensionar los tamaños de la ventana. En la mayoría de los casos se puede configurar.
* **Cabecera**: Es dónde se encuentran situados los iconos que identifican y permiten llegar a las principales funcionalidades de la aplicación que se ejecuta en esa ventana.
* **Espacio del contenido:** Varía de tamaño en función de las dimensiones de la ventana. Es dónde se muestra la información (imagen, contenidos multimedia, de texto...).
* **Barra de desplazamiento (scroll):** Aparece en caso de que el espacio del contenido sea menor que la información que se va a mostrar. De esta forma el usuario se puede desplazar por los contenidos, en principio, no visibles de la ventana.
* **Pie de la ventana:** Se utiliza para ofrecer informaciones diversas relacionadas con la ventana y/o aplicación y los contenido que muestra.

Los **cuadros de diálogo (ventanas secundarias)** son un tipo de ventanas, también consideradas como ventanas secundarias. Aparecen por encima del resto de ventanas pidiendo una interacción concreta al usuario, que deberá contestarle.

La interacción puede ser sólo informativa (una advertencia que se tiene que confirmar para poder continuar) o un requisito concreto de algún dato o de alguna acción que habrá que contestar para poder continuar utilizando la aplicación.

**Ejemplo**: la ventana de finalizar una aplicación o de borrar un archivo. Este cuadro de diálogo no nos permite seguir trabajando con esa aplicación hasta que no haya contestado su mensaje.

Los **asistentes** son otro tipo de ventanas que van apareciendo una detrás de otra hasta que se alcanza un determinado objetivo.

La idea de este tipo de ventanas es que el usuario no se olvide de introducir una información y que dentro de lo posible pueda escoger las opciones que más le interesen.

**Ejemplo**: las ventanas que aparecen cuando instalas un programa, en las que preguntas paso a paso todo lo que se necesita.

Los **menús** son un elemento muy importante de las GUI ya que se encargan de mostrar todas las posibilidades de interactuar con las aplicaciones.

Cada opción del menú dispondrá de otro conjunto de opciones → árbol jerarquizado con un número de niveles determinado. Cuanto más niveles tenga, más complicado será que un usuario llegue a una determinada funcionalidad.

Los menús han ido evolucionando de forma que ahora muestran las funcionalidades por medio de iconos, minimizando la presencia de texto.

Los menús se encuentran normalmente justo debajo del marco superior. La forma más habitual de seleccionar una opción de un menú será utilizando el ratón (haciendo click se activará y mostrará todas las opciones relacionadas) → Además deben ser accesibles para otros dispositivos de entrada (teclado) por si el usuario no dispone de un ratón. En ocasiones los menús nos indican cuales son las otras opciones (ctrl +, F11) para llegar a una determinada funcionalidad.

Existen diferentes tipos de menús, dependiendo de su entorno de ejecución o de la tipología de aplicación a la que pertenece GUI. Los más comunes son los siguientes:

* **Menús jerárquicos**: Se representan en forma de árbol. Consta de niveles horizontales (opciones del menú) y de niveles verticales (hasta dónde podrá llegar el usuario navegando por el menú). Se suelen utilizar en SO.
* **Menús contextuales**: No se encuentran visibles en la interfaz hasta que el usuario no los activa. Se abren en una ventana flotante que varía en función de dónde esté el ratón a la hora de llamarlos (haciendo click derecho).
* **Menús de navegación (scrolls):** Evolución de los menús jerárquicos. Sólo muestra las opciones más utilizadas de los menús, “ocultando” el resto, y tienen un pequeño icono que al seleccionarlo muestra el resto de las opciones. Se utiliza cuando se quieren mostrar muchas opciones en una interfaz de usuario.

Las **pestañas (pestañas de propiedades)** son las que permiten mostrar un conjunto de datos y/o funcionalidades relacionadas entre sí de forma agrupada. En una misma ventana se muestran los contenidos de tantas ventanas como se quiera, separándolas en pestañas identificadas por un título y que contiene información relacionada.

Las **barras de herramientas** se caracterizan por facilitar el acceso a determinadas funcionalidades mediante un icono. Todas estas funcionalidades se pueden encontrar en los menús, pero no todo lo que ofrece el menú aparece en la barra de herramientas.

Normalmente en la barra de herramientas se ponen las funcionalidades que más se utilizan. Las combinaciones de teclas son otro método para acceder a estas opciones de los menús.

La idea es facilitar a una sola acción (un único click) el acceso a las principales funcionalidades → son más sencillas y accesibles.

Los **iconos** son imágenes que representan funcionalidades o acciones que se podrán realizar realizando click encima. En las GUI se suelen utilizar normalmente en las barras de herramientas.

Permiten identificar archivos, carpetas u otras aplicaciones o dispositivos de un sistema informático. Tienen que ser fácilmente identificables por los usuarios, en caso de que no sea así estaríamos obligando al usuario a memorizar cierta información para acceder a las funcionalidades principales.

Es un elemento importante de las GUI ya que hace que sea más rápida, atractiva y sencilla el utilizar la aplicación. Además de que permite con mayor facilidad que las aplicaciones sean internacionales.

Los **entornos de trabajo** son un elemento más de una GUI. Es la ubicación principal de las informaciones con las que interactúa el usuario, dónde se muestra el texto, imágenes, o datos que el usuario ha solicitado a través de los menús.

Los **controles** son los que proporcionan funciones a las GUI consiguiendo así muchas más interacciones entre interfaz y usuario. Un tipo de control son los botones, objetos de control que dan la opción de introducir un dato de confirmación en el sistema. Existen diferentes tipos de botones:

* **Botones en forma de radio (radio buttons)**: se utilizan en formularios o menús para dar la posibilidad al usuario de seleccionar una (única) opción de una lista. Suelen ser redondos.
* **Botones de confirmación (check box)**: similares a los de radio y que normalmente tienen forma cuadrada. También se usan para seleccionar una o varias opciones de la lista.
* **Botones de relieve**: imitan a un botón de un dispositivo físico. Simulan que tienen volumen y por tanto pueden tener varios estados (activado o no, seleccionado o no). Incluye un texto que representa la funcionalidad del mismo.

Los **controles** no sólo son los botones, también son los elementos de entrada de texto o de información de salida, las barras de progreso o de estado, el combo box etc.

Las **tipografías** son la forma en la que muestra el texto que se utiliza en las opciones de los distintos elementos de las GUI. La elección de estas tipografías también ofrece un mensaje específico al usuario y por tanto se debe tener en cuenta en el diseño y desarrollo de las GUI.

La *tipografía digital* son los tipos de letra desarrollados exclusivamente para pantallas, para su uso en GUI de aplicaciones informáticas. Normalmente cada SO tiene desarrollada su tipografía.

Una de las características de las tipografías son la intensidad, los colores y los tamaños utilizados en las letras. La elección de esto, influye bastante en la apreciación por parte del usuario de la GUI desarrollada.

**7. Eventos y diálogos**

Un **cuadro de diálogo** es una ventana que aparecerá ofreciendo una información a los usuarios en un momento determinado de la ejecución de una interfaz.

Decimos que un cuadro de diálogo es **modal** cuando la aplicación misma o el formulario que le ha llamado no puede recibir ningún evento hasta que se haya cerrado el cuadro de diálogo. Los que necesitan que el usuario les confirme de alguna forma que lo han visto.

Decimos que un cuadro de diálogo es **no modal** cuando la aparición del cuadro de diálogo no limita al usuario a continuar interactuando con el resto de ventanas.

Un **cuadro de mensaje** es un cuadro de diálogo modal que puede utilizarse para mostrar información textual y permitir que los usuarios tomen decisiones con botones.

Para crear un cuadro de mensaje, se utiliza la clase *MessageBox*.

Otro tipo de cuadros de diálogo son los cuadros de diálogo comunes, que pueden ser tan modales como no modales. → Windows implementa varios cuadros de diálogo **reutilizables** que son comunes para todas las aplicaciones, por ejemplo cuadros de diálogo para abrir archivos, guardar archivos e imprimir. →. Un ejemplo de cuadro de diálogo sería el creado cuando se intenta imprimir un documento.

Una aplicación no sabe cuando un usuario quiere realizar una funcionalidad. Los **eventos** son los proporcionan los mecanismos adecuados para que se produzca esa interacción (de forma asíncrona) durante la ejecución de un programa. Normalmente son producidos desde el exterior de la aplicación, por ejemplo cuando se hace click en algo o cuando se presiona una tecla.

No tenemos que tratar todos los eventos, solo lo aplicaremos en aquellos que nos interese.

Algunos de los eventos que más se utilizan en las GUI son:

* **MouseMove**: al mover el ratón por encima de un control.
* **MouseDown**: al pulsar cualquier botón del ratón.
* **Change**: al cambiar el contenido del control.
* **Click**: al hacer click con el botón izquierdo del ratón sobre el control.
* **GetFocus**: este evento se activa cuando el control recibe el enfoque, es decir, cuando se activa el control en tiempo de ejecución para introducir datos o realizar alguna operación.
* **LostFocus**: es lo contrario del evento anterior, se activa cuando el control pierde el enfoque, es decir, se pasa a otro control para seguir introduciendo datos.