UT 1 – Diseño de Interfaces de usuario.

Objetivos

- ✓ Reconocer la importancia de la comunicación visual y sus principios básicos.
- ✓ Identificar y analizar los elementos para la elaboración de prototipos.
- ✓ Planificar el proceso de elaboración del diseño de una interfaz.
- ✓ Analizar los distintos entornos de desarrollo y escoger el que más se adecua al diseño de cada proyecto.

ÍNDICE

María del Cristo García León.

Departamento Informática y Comunicaciones.

IES El Rincón.

1. Interacción persona – ordenador.

Lo primero que debemos plantearnos a la hora de diseñar una interfaz es *¿qué es la interacción persona-ordenador?* Podemos encontrar multitud de definiciones, pero nos vamos a quedar con que es la disciplina que estudia el intercambio de información entre las personas y los ordenadores, cuyo objetivo es que este intercambio sea más eficiente, es decir, disminuyan los errores, se incremente la satisfacción, etc.

Licklider y Clark, en 1962, elaboran una lista con los diez problemas más comunes que deberán ser resueltos para facilitar la interacción entre las personas y los ordenadores:

1. Compartir el tiempo de uso de los ordenadores entre muchos usuarios.
2. Un sistema de entrada-salida para la comunicación mediante datos simbólicos y gráficos.
3. Un sistema interactivo de proceso de las operaciones en tiempo real.
4. Sistemas para el almacenamiento masivo de información que permitan su rápida recuperación.
5. Sistemas que faciliten la cooperación entre personas en el diseño y programación de grandes sistemas.
6. Reconocimiento por parte de los ordenadores de la voz, de la escritura manual impresa y de la introducción de datos a partir de escritura manual directa.
7. Comprensión del lenguaje natural, sintácticamente y semánticamente.
8. Reconocimiento de la voz de varios usuarios por el ordenador.
9. Descubrimiento, desarrollo y simplificación de una teoría de algoritmos.
10. Programación heurística o a través de principios generales.

Hansen (1971) en su libro *User Engineering Principles for Interactive Systems* hace la primera enumeración de principios para el diseño de sistemas interactivos:

1. Conocer al usuario.
2. Minimizar la memorización, sustituyendo la entrada de datos por la selección de ítems, usando nombres en lugar de números, asegurándose un comportamiento predecible y proveyendo acceso rápido a información práctica del sistema.
3. Optimizar las operaciones mediante la rápida ejecución de operaciones comunes, la consistencia de la interfaz y organizando y reorganizando la estructura de la información basándose en la observación del uso del sistema.
4. Facilitar buenos mensajes de error, crear diseños que eviten los errores más comunes, haciendo posible deshacer acciones realizadas y garantizar la integridad del sistema en caso de un fallo de software o hardware.

2. Diseño de una interfaz. El diseño gráfico.

El diseño gráfico consiste en programar, proyectar y realizar comunicaciones visuales de aplicaciones u otro tipo de herramientas software que generalmente serán transmitidos por medios industriales. A día de hoy, esta área de desarrollo tiene a sus propios profesionales, denominados *diseñadores gráficos*. Se destacan sus tres grandes funciones:

- Función estética.
- Función publicitaria.
- Función comunicativa.

Se distinguen cuatro grupos de elementos en el diseño de interfaces: elementos conceptuales, elementos visuales, elementos de relación y elementos prácticos.

Por lo tanto, la interfaz será un conjunto de elementos gráficos y un diseño de su distribución que permite una mejor presentación y navegación a través de la aplicación. Si no existen ambos factores unidos, si el resultado final de nuestro sitio es óptimo, será fruto de la casualidad. Podemos ver en las siguientes imágenes dos diseños, uno bueno y otro no tanto.

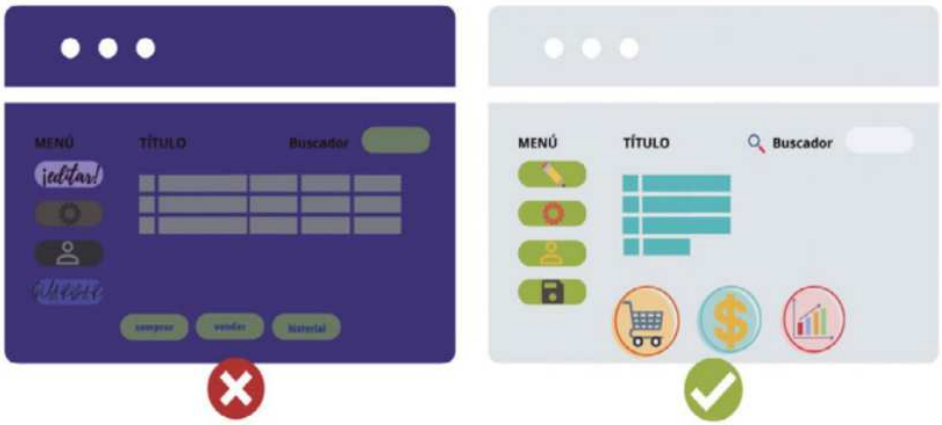


Figura 1.1
Diseños de contraejemplo: un buen y un mal diseño gráfico.

Como se puede ver en la figura 1.1, aparecen contrapuestas dos interfaces para una misma aplicación, en la de la izquierda podemos observar un "mal" diseño, se trata de una aplicación en la que los elementos no aparecen claramente diferenciados, no se puede leer bien el texto, no se aprecian bien las imágenes, etc. Mientras que, en la aplicación de la derecha, la navegación del usuario resulta mucho más intuitiva, proporcionando un mayor grado de satisfacción, lo que se debe, entre otros factores, a que los elementos aparecen claramente dispuestos y no hay sobrecarga de estos.

En la construcción de cualquier aplicación y su interfaz correspondiente, debemos tener en cuenta diferentes fases, desde la definición de los objetivos que se persiguen con nuestro proyecto hasta el diseño visual resultante, pasando por las especificaciones funcionales, entre otros.

En la siguiente pila, vamos a definir todas estas fases que se deben tener presentes en el desarrollo de la interfaz.

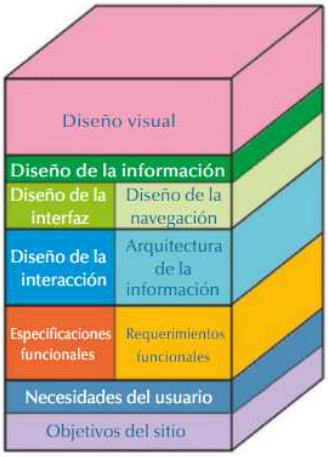


Figura 1.2
Planos de jerarquía de diseño de una aplicación.

Tras identificar las tareas que debe realizar una aplicación, en primer lugar definiremos los objetos y las acciones; este enfoque es similar a la definición de clases en un diseño orientado a objetos. Lo primero que se aconseja es realizar un modelado textual del escenario de la aplicación, distinguiendo entre sustantivos y verbos, los primeros defienden los objetos y elementos, y los segundos las acciones sobre los anteriores. Es habitual diferenciar los siguientes tipos de objetos:

a) *Objetos origen:* elementos de la aplicación que permiten ser desplazados a cualquier otro punto, siendo depositados sobre un objeto destino.
b) *Objetos destino:* elementos que reciben los objetos origen.
c) *Objetos de la aplicación:* elementos propios de la aplicación que no permiten una interacción directa con el usuario.

Uno de los aspectos más importantes en cuanto al diseño de interfaces recae sobre el tipo de la pantalla donde será mostrado, se ha de analizar por completo el diseño gráfico que vamos a emplear, la colocación de los iconos, el dispositivo de interacción, los textos descriptivos, etc. Una vez finalizado el diseño, debemos realizar una revisión del mismo para descubrir los posibles problemas de funcionamiento y su corrección.

- ✓ *Tiempo de respuesta del sistema.* El tiempo de respuesta hace referencia a dos conceptos clave: duración y variabilidad. El primero queda definido como el tiempo que se emplea para completar una acción. Escoger este valor no es trivial, puesto que si es muy corto el usuario puede no haber completado la acción, pero si es demasiado largo, puede suponer su desistimiento.
- ✓ *Servicios de ayuda al usuario.* Existen dos tipos de ayuda al usuario: ayudas integradas en la aplicación, que aparecen en la mayoría de casos de forma automática, como sugerencia o guía al iniciar una tarea; y ayudas complementarias, que normalmente se encuentran en forma de manuales disponibles en la propia herramienta o a través de Internet. Estos manuales permiten filtrar el contenido que se busca a través de la generación de pequeñas consultas.
- ✓ *Etiquetado de órdenes.* Este aspecto se centra en el correcto diseño de la nomenclatura asociada a cada acción, es decir, que el elemento textual o visual correspondiente a una determinada función sea una palabra adecuada para cada acción, así como incluir atajos de teclado significativos y fáciles de recordar, o emplear teclas de función para las órdenes más usadas.

3. Herramientas.

El primer paso para el diseño de una interfaz es la elaboración de una maqueta o prototipo, la consecución de una buena versión previa de lo que más adelante se va a desarrollar mejora la velocidad de desarrollo de la aplicación. Existen diferentes tipos de herramientas que nos permiten llevar a cabo una adecuada construcción de prototipos, incluyendo ventanas, menús, tratamiento de errores, cuadros de diálogo, etc.

Uno de los puntos más importantes para definir un buen diseño es la evaluación del mismo, el diseño de esta puede resultar determinante. Se aconseja al menos llevar a cabo el siguiente proceso:

1. A través de un formulario de carácter cuantitativo o cualitativo se recoge el grado de satisfacción del usuario. En el segundo de los casos será posible evaluar el tiempo empleado por el usuario para aprender el funcionamiento de la aplicación, el tiempo de lectura de la ayuda, etc.
2. Gracias a los datos recogidos, el diseñador puede comprobar si la interfaz cumple los requisitos planteados al comienzo del diseño.
3. Si el análisis anterior no cumple las expectativas de diseño, será necesario revisar de nuevo el mismo para que se adecue a las directrices iniciales, así como para solventar los errores o defectos que se hubieran podido detectar. Este proceso se repetirá hasta que se cumplan los requisitos previos de diseño para la interfaz.

Las herramientas disponibles para el desarrollo de entornos gráficos pueden ordenarse en una primera clasificación entre comerciales y libres. Las primeras son las desarrolladas por empresas para su venta; dentro de esta categoría encontramos Microsoft y Borland.

3.1 Prototipos. Elementos clave de un prototipo.

Un prototipo consiste en una maqueta o modelo de un diseño para que nos hagamos una idea del producto final que se obtendrá. El prototipo nos permite ver el resultado de distintos diseños finales, comprobar alguna funcionalidad o realizar test de usabilidad. De esta forma, ahorramos tiempo, esfuerzo o dinero, puesto que es más sencillo realizar cambios sobre un diseño previo y no sobre un producto final.

Aplicando el uso de prototipos al diseño de interfaces, su implementación es fundamental, puesto que permite realizar multitud de diseños previos de los menús, elementos y demás partes del diseño para adecuarlos a las necesidades del cliente, antes de comenzar a escribir el código.

Algunas de las ventajas que justifica el uso del prototipo son:

- *Mejora la velocidad de desarrollo.* Es más eficiente realizar los cambios sobre un prototipo antes de comenzar su desarrollo que sobre un diseño definitivo, en el que aparezcan colores y tipografías, puesto que el cliente perderá la atención en el diseño base, que es el objeto principal del diseño de interfaces y del prototipo como primer paso de desarrollo.
- *Involucra al cliente.* Como se ha comentado en el punto anterior, el cliente es el responsable de aprobar el diseño último del sitio, por lo tanto, será mucho más sencillo y útil proponer y hacer cambios sobre un esquema con poco detalle que sobre un diseño ya acabado. Es fácil involucrar al cliente en esta fase.

No existe un tipo único de esquemas de prototipo, sino que encontramos múltiples, como esquemas de página, wireframes, prototipos, mockups, bocetos, sketches, diagramas. Se pueden distinguir tres tipos, tal y como indica Daniel Torres Burriel:

- ✓ *Sketching:* para dibujar toda la interfaz de la aplicación, los procesos y las relaciones entre pantallas (solo papel). Este primer tipo es el que se suele emplear en la fase inicial, donde realizaremos diseños esquemáticos en papel. Se basa esencialmente en establecer la jerarquía de contenidos, pero sin detalles de diseño.
- ✓ *Wireframing:* se utiliza para dibujar con un cierto nivel de detalle las pantallas, los esbozos de contenido, las llamadas a la acción y, en general, la disposición física de los elementos (papel o digital). En esta fase se desarrolla y entrega una maqueta con base en lo "diseñado" en el paso previo. De esta forma es posible validar los aspectos de diseño por parte del cliente. Lo fundamental en esta fase es la organización de los contenidos.
- ✓ *Prototipado:* para diseñar y ejecutar la interacción entre las pantallas que componen los procesos (solo digital). El prototipado se utiliza como paso final, puesto que permite evaluar no solo el diseño y la organización, sino también el funcionamiento, la interacción (menús, formularios, botones, iconos). Se emplea para hacer pruebas de usuarios antes de tener hecho el desarrollo completo e implantado del producto. Esto ahorra horas de desarrollo, ya que son necesarias menos versiones de la aplicación. Podemos encontrar tres clases de prototipos basados en su funcionalidad:

a) *Horizontal*. Modela muchas características de una aplicación, pero incorporando pocos detalles. Es el prototipo utilizado en las primeras etapas de diseño.
b) *Vertical*. Modela pocas características, pero en este caso se añaden más detalles.
c) *Diagonal*. Se trata de un prototipo mixto entre los dos anteriores. Hasta cierto nivel presenta las características del tipo horizontal, a partir del cual implementa las del tipo vertical.

3.2 Aplicaciones para el desarrollo de interfaces.

El diseño del prototipo, en cualquiera de sus fases, se debe basar en los siguientes aspectos:

- Identificar los elementos que forman parte de cada una de las pantallas de una aplicación.
- Distribuir el número de elementos de la interfaz gráfica para que no exista saturación de elementos, pero haya suficiente información en la misma y la interacción sea correcta.
- Organización de la jerarquía de elementos, orden y disposición de los mismos.
- Extensión adecuada del diseño para aprovechar eficientemente el espacio en función del dispositivo.
- Patrones de diseño para estandarizar el de interfaces.
- Aspectos técnicos de usabilidad y accesibilidad.

Cuando hablamos de una interfaz gráfica, podemos diferenciar claramente dos áreas, aquella común a todas las ventanas de una misma aplicación y, por otro lado, la parte de contenido que varía de una ventana a otra.

PARA SABER MÁS

La *jerarquía visual* es la disposición de los elementos. No se trata de algo trivial, sino que es necesario definir de forma eficiente las prioridades de comunicación, información e interacción.

Los elementos se deben situar de izquierda a derecha y de arriba hacia abajo para establecer una jerarquía que va de mayor a menor importancia. Lo más importante va arriba a la izquierda, y va perdiendo fuerza o importancia aquello que se relega hacia abajo y a la derecha.

El uso de herramientas basadas en componentes visuales para el desarrollo de interfaces presenta numerosas ventajas, pero una de las más importantes es la facilidad con la que se pueden empaquetar los componentes desarrollados para ser reutilizados posteriormente. La reutilización del código es un aspecto clave para la implementación de nuevos proyectos, puesto que simplifica el desarrollo y permite prestar atención a aspectos más importantes y no repetir fragmentos de código previamente desarrollados y probados. Algunas de las herramientas más conocidas en la actualidad se muestran a continuación:

- ✓ *Visual Studio.* Entre las fortalezas más importantes de este IDE se encuentra el uso de lenguajes multiplataforma. Se puede escribir en los lenguajes C#, F#, Razor, HTML5, CSS, JavaScript, Typescript, XAML y XML.

  Este entorno de desarrollo incorpora la funcionalidad de autocompletado de código, lo que permite detectar los problemas en tiempo real. A la hora de depurar el código, permite ir paso a paso, estableciendo puntos de interrupción, por procedimientos y por instrucciones. Por último, cabe destacar que permite administrar el código en los repositorios más utilizados en la actualidad, como son GIT, GitHub y Azure DevOps. De esta forma es posible controlar las modificaciones entre las diferentes versiones de nuestros proyectos y poder realizar copias de seguridad de los archivos durante el desarrollo del mismo.
- ✓ *Monodevelop.* Este IDE es libre y gratuito, incorpora todas las funcionalidades propias de un editor de texto además de otras que permiten depurar y gestionar proyectos. Entre sus principales ventajas se encuentran que permite trabajar con algunos de los lenguajes más demandados en la actualidad, como son C#, Java, .NET y Python. Pertenece a Unity, motor de videojuegos multiplataforma por excelencia.

- ✓ *Glade.* Permite la creación de interfaces gráficas de usuario. Es muy utilizada en entornos XML. Permite el desarrollo de interfaces gráficas basadas en lenguaje C, C++, C#, Java, Python.
- ✓ *NetBeans.* Herramienta de código abierto. Se trata de uno de los entornos de desarrollo más utilizados en cuanto al desarrollo de interfaces a través de lenguaje de programación en Java. Este IDE permite extender el entorno con un gran número de módulos que agrupan clases de Java que permiten interactuar con las API de NetBeans.
- ✓ *Eclipse.* Entorno de desarrollo de código abierto y multiplataforma. Dispone de la funcionalidad Graphical Layout que nos permite visualizar el contenido en vista de diseño y desarrollar componentes visuales de una forma rápida e intuitiva. Cabe destacar su componente Palette, un panel que permite crear botones, cuadros de texto, cuadrículas, insertar imágenes.

Comparativa de las herramientas de edición de interfaces

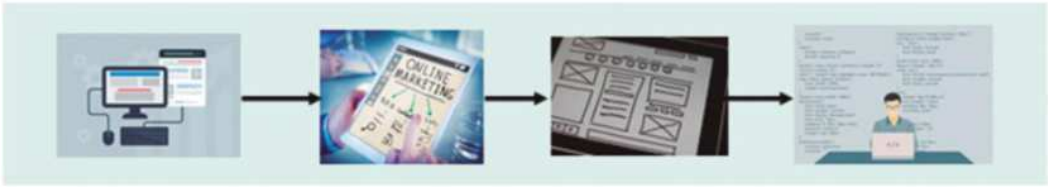
Nombre	Licencia	Lenguajes soportados	Enlace
Visual Studio *Community*	Propietaria *Libre*	C#, HTML, Javascript, XML	https://visualstudio.microsoft.com/es/
Mono Develop	Libre	C#, Java, .NET, Python	https://www.monodevelop.com
Glade	Libre	C++, C#, Java, Python	https://glade.gnome.org
NetBeans	Libre	Java, HTML, PHP, Python	https://netbeans.org
Eclipse	Libre	Java, C++, PHP	https://www.eclipse.org

4. Planteamiento y diseño de una interfaz.

Para finalizar este primer capítulo de introducción, en el que ya hemos visto la importancia del diseño en las interfaces, así como algunas herramientas que nos pueden resultar útiles, nos centramos en los elementos clave que debemos tener en cuenta para crear el prototipo de nuestra interfaz.

a) Los elementos que van a formar parte de nuestra aplicación. Debe haber un número suficiente, pero sin que haya saturación de estos.
b) La extensión de la aplicación.
c) Patrones de diseño que van a utilizarse para estandarizar el diseño de interfaces.
d) Aspectos técnicos de usabilidad y accesibilidad.

Pasos para el diseño de un sitio web:



Definición del propósito final | Necesidades y requisitos | *Wireframing* | Desarrollo e implementación (prototipo)

4.1 Área de redacción.

El área de redacción es la encargada de delimitar los pilares fundamentales de los que debe constar el proyecto que se está desarrollando. El aspecto principal que se debe establecer es el objeto final que busca nuestra aplicación. Por ejemplo, no es lo mismo realizar el diseño de una aplicación que se va a dedicar a vender respuestas de neumáticos, que una aplicación para la venta de entradas de conciertos.

4.2 Área de producción.

Una vez que se ha definido el propósito final del proyecto, es necesario que este se encuentre en consonancia con las circunstancias del mercado actual, llevar a cabo un estudio de la viabilidad y trazar el plan de desarrollo más adecuado. Los agentes que participan en la producción son:

- *Cliente.* Se trata de quién encarga la creación y desarrollo de la aplicación del proyecto.
- *Usuario.* El público hacia el que va dirigida la aplicación. Son los receptores finales del proyecto, por lo tanto, es interesante hacer un estudio de estos, conocer sus necesidades y demandas, con el fin de mejorar el rendimiento de la aplicación.
- *Presupuesto.* Se trata del total económico que el cliente desea destinar a la construcción de la aplicación. En base a este se escogerá el gestor de contenidos, entre otros elementos.
- *Plan de trabajo.* Calendario de entregas, distribución de las tareas, etc.

4.3 Área técnica.

Esta área se encuentra constituida por los responsables técnicos, que son los encargados de realizar un estudio de los requisitos del proyecto relativos a su programación, normalmente en HTML, CSS y bases de datos. Hasta aquí el resto de las áreas se han basado en uso de *sketching*, en un primer momento, y en *wireframing*. El área técnica comienza el desarrollo de los primeros prototipos y maquetas. Recordemos que estos ya comienzan a incorporar los primeros elementos de interacción, hasta llegar al producto final.

4.4 Área artística.

Esta última área se centra en la estética final del proyecto, en base al propósito final de uso de la aplicación, así como a los informes de usuario y los requisitos del cliente. Hay que tener en cuenta que el estilo variará en función de la aplicación y de los usuarios. El estilo del proyecto se debe convertir en una seña de identidad, en esto consiste realizar un buen diseño gráfico, dotar de personalidad propia a una aplicación y que esta sea fácilmente reconocible en cualquier situación.

---

5. El color.

El ojo humano solo es capaz de percibir los denominados *colores aditivos*; a través de la combinación de estos le es posible obtener el resto de los colores, esto son: azul (B), rojo (R) y verde (G). En este apartado analizaremos en qué consiste el sistema de representación RGB, así como las propiedades principales del color, que modifican y redefinen el sistema de color base.

5.1 Sistema RGB.

De la misma forma, un ordenador será capaz de obtener la representación de todos los colores utilizando el Sistema RGB, o lo que es lo mismo, el Sistema Red-Green-Blue. Indicando la proporción de cada uno de ellos dentro de la combinación de estos tres, dará lugar a toda la paleta de colores conocida.

Para representar cada color de forma que pueda ser traducido por el ordenador se utilizan 8 bits para codificar cada uno de los colores aditivos, es decir, se establece la proporción de cada color que va a formar parte de la combinación de tres. La escala monocromática de un color tendrá 256 (2^8) valores.

A la hora de representar cada uno de los colores, es posible utilizar tanto el sistema de numeración decimal (0 a 255) como el hexadecimal, donde cada uno de los dígitos se codifica con 8 bits binarios que, agrupados en bloques de 4 bits, nos devuelve el valor correspondiente en hexadecimal.

El número de combinaciones de colores se calcula multiplicando el número máximo de grados en la escala monocromática de cada color, 256 × 256 × 256, lo que nos da 16 777 216 colores.

Figura 1.5
Círculo de colores aditivos en Sistema RGB y combinación de ellos.

Ejemplo

El color amarillo estaría formado por:

Rojo = 255	→	1111 1111	→	f f
Verde = 255	→	1111 1111	→	f f
Azul = 0	→	0000 0000	→	0 0

En hexadecimal queda expresado por: #ffff00

5.2 Matiz, saturación y brillo.

Además del grado en la escala monocromática de cada uno de los colores del sistema RGB, los colores presentan tres propiedades que permiten distinguirse unos de otros, estas son: el matiz, la saturación y el brillo. Estas propiedades nos permiten definir los colores como cromáticos, complementarios o cercanos, así como definir el contraste de color.

- ✓ *Matiz*. Atributo que permite distinguir un color de otro. Los tres matices primarios son los colores aditivos, verde, rojo y azul; el resto de colores se obtiene mezclando estos tres. El matiz permite definir dos colores como complementarios cuando está uno frente al otro en el círculo cromático.
- ✓ *Saturación*. Este atributo define la intensidad de un color. Puede relacionarse con el ancho de banda de luz que estamos visualizando, por lo tanto, queda condicionado por el nivel de gris presente en un color: cuanto mayor sea el nivel de gris, menos saturado será un color, y será menos intenso.

Figura 1.6
Círculo cromático de matices.

Figura 1.7
Escala de color con matiz rojo modificando sus valores de saturación.

- ✓ *Brillo*. Atributo que define la cantidad de luz de un color. Representa lo oscuro (si se le añade negro) o claro (si se le añade blanco) que es un color respecto de su patrón, es decir, respecto del color puro sin modificar el brillo. En una composición de colores en diseño gráfico, cuanto más brillante sea un color, más cerca parece estar.

Figura 1.8
Escala de color con matiz rojo modificando sus valores de brillo.

TOMA NOTA

En el diseño de interfaces gráficas, la selección adecuada de la carta de colores es muy importante, puesto que esto puede condicionar la experiencia de navegación del usuario, determinándola por completo. La opción más sencilla es escoger la *monocromía*, que consiste en elegir un solo color del círculo cromático y, a continuación, obtenemos su variedad de tonalidades, añadiendo blanco o negro.

Otra de las alternativas más utilizadas son los denominados *colores vecinos*. Estos son armónicos, que proporcionan estabilidad en el diseño de colores. Se denominan *vecinos* aquellos colores que se encuentran en un rango de 90º en el círculo cromático.

Resumen

- La interacción persona-ordenador se centra en el estudio del intercambio de información entre las personas y los ordenadores, y su objetivo es que este intercambio sea más eficiente, es decir, disminuyan los errores, se incremente la satisfacción, etc. En el libro *User Engineering Principles for Interactive Systems* de Hansen (1971) se lleva a cabo una enumeración de principios para el diseño de sistemas interactivos que perdura hasta la actualidad:

  1. Conocer al usuario.
  2. Minimizar la memorización.
  3. Optimizar las operaciones mediante la rápida ejecución de operaciones comunes.
  4. Facilitar buenos mensajes de error, crear diseños que eviten los errores más comunes, haciendo posible deshacer acciones realizadas y garantizar la integridad del sistema en caso de un fallo de software o hardware.

- La base de una adecuada interacción está en el desarrollo de un correcto diseño gráfico. Este consiste en programar, proyectar y realizar comunicaciones visuales de aplicaciones, u otro tipo de herramientas software que generalmente van a ser transmitidas por medios industriales. Se destacan sus tres grandes funciones: función estética, función publicitaria y función comunicativa.
- Para comenzar el desarrollo de un proyecto de diseño de interfaz, en primer lugar, se realizan maquetas o modelos de diseño previos para crear una idea inicial del producto final, comprobar alguna funcionalidad, etc. Existen múltiples tipos de prototipos, aunque nos centramos en los tres más comunes: *sketching,* para dibujar toda interfaz de la aplicación, los procesos y las relaciones entre pantallas (solo papel); *wireframing,* fase en la que se desarrolla y entrega una maqueta en base en lo "diseñado" en el paso previo; y *prototipado,* que se utiliza como paso final, puesto que permite evaluar no solo el diseño y organización, sino también el funcionamiento, la interacción.
- Tras identificar las tareas que debe realizar una aplicación, primero definiremos los objetos y las acciones. Este enfoque es similar a la definición de las diferentes clases en un diseño orientado a objetos. Lo primero que se aconseja es realizar un modelado textual del escenario de la aplicación, distinguiendo entre sustantivos y verbos: los primeros defienden los objetos y elementos, y los segundos las acciones sobre los anteriores. Es habitual diferenciar los siguientes tipos de objetos:

  - *Objetos origen:* elementos de la aplicación que permiten ser desplazados a cualquier otro punto, siendo depositados sobre un objeto destino.
  - *Objetos destino:* elementos que reciben los objetos origen.
  - *Objetos de la aplicación:* elementos propios de la aplicación que no permiten una interacción directa con el usuario.

- El uso de herramientas basadas en componentes visuales para el desarrollo de interfaces presenta numerosas ventajas, pero una de las más importantes es la facilidad con la que se pueden empaquetar los componentes desarrollados para ser reutilizados posteriormente. La reutilización del código es un aspecto clave para la implementación de nuevos proyectos puesto que simplifica el desarrollo y permiten prestar atención a aspectos más importantes y no repetir fragmentos de código previamente desarrollados y probados. Algunas de las herramientas más conocidas en la actualidad son: Visual Studio, Monodevelop, Glade, NetBeans o Eclipse.
- El ojo humano solo es capaz de percibir los denominados colores aditivos; a través de la combinación de estos le es posible obtener el resto de los colores, esto son: azul (B), rojo (R) y verde (G). De la misma forma, un ordenador será capaz de obtener la representación de todos los colores utilizando el Sistema RGB, o lo que es lo mismo, el Sistema Red-Green-Blue. Indicando la proporción de cada uno de ellos dentro de la combinación de estos tres, se dará lugar a toda la paleta de colores conocida. Para representar cada color de forma que pueda ser traducido por el ordenador se utilizan 8 bits para codificar cada uno de los colores aditivos, es decir, se establece la proporción de cada color que va a formar parte de la combinación de tres. La escala monocromática de un color tendrá 256 (2^8) valores.