# Usabilidad. Principios.

1. **Adoptar el punto de vista del usuario**. El usuario deberá poder usar la aplicación sin necesidad de un manual. Debe ser intuitivo.
2. **Realimentación**: barras de progreso, resaltar la opción elegida del menú, comunicar el éxito o fracaso de una operación, etc.
   1. Esto implica usar hilos u otros mecanismos para atender estas tareas, que son operaciones que se estén realizando mientras el usuario ve su progreso.
   2. Es muy común ver una ProgressBar en la barra de estado.
3. **Consistencia y homogeneidad**: los menús deben seguir siempre el mismo orden (Archivo, Editar, Ver, etc.) en todas las interfaces del programa (y también para parecerse a otros programas ya existentes y que el usuario pueda generalizar sus conocimientos de un aspecto de la interfaz a otro).
   1. **También otro aspecto importante asociado a la homogeneidad es que todo esté en el mismo idioma.** Si el programa tiene internacionalización, pero las traducciones no están al 100% en cada uno de los idiomas, tendremos un programa poco consistente y homogéneo.
4. **Tratamiento de errores**: minimizar la posibilidad de cometer errores.
   1. Uso de spinners

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

* 1. Inhibir controles que representan acciones no permitidas.
  2. Pedir confirmación de operaciones peligrosas.
  3. Evitar los controles que sean físicamente difíciles de manejar (controles muy pequeños, menúes emergentes o submenúes con muchas opciones).
  4. En caso de error, dar información suficiente, pero adaptada al punto de vista del usuario (tanto a sus objetivos como a sus conocimientos).

|  |  |
| --- | --- |
| **ASÍ SI** | **ASÍ NO** |
|  |  |

* 1. Facilitar la recuperación en caso de error. Por ejemplo: Ctrl + Z, Ctrl + Alt + Supr, etc.

1. **Minimizar la necesidad de memorización**:
   1. Usar controles gráficos (spinners, cosas así) en lugar de campos de texto (TextField) todo el rato.
   2. Utilizar nombres y símbolos autoexplicativos y fáciles de recordar.

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza media

* 1. **Minimalismo**: limitar la carga de información a corto plazo. Tener solo la información necesaria en pantalla (reglas de “menos es más” y del 7+-2). Por ejemplo: la interfaz Ribbon de Microsoft agrupa controles en grupos de como máximo, 7.



Interfaz de usuario gráfica, Aplicación, Tabla

Descripción generada automáticamente

1. **Diseñar diálogos para conducir a la finalización**. Por ejemplo, los asistentes tipo Wizard. En general, se debe intentar minimizar el nº de decisiones que el usuario ha de tomar. Así que cuidado con hacer asistentes larguísimos.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

1. **Usar metáforas e invitaciones**:
   1. **Metáforas:** usar mecanismos y procesos conocidos por el usuario, generalmente en la vida real. Por ejemplo: el uso del icono del disquete, para guardar archivos (aunque ya no se use el disquete, pero en su momento sí, y esto se ha convertido en un símbolo estándar), o el escritorio de Windows, que representa una “oficina virtual” con sus iconos (carpetas, archivos y papelera).
   2. **Invitaciones**: por ejemplo, que el cursor cambie de forma al pasar sobre un enlace, o que un botón se “ilumine” (hover) son invitaciones.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

**Aspectos más secundarios**:

1. **Internacionalización**: no usar strings hard-coded. Es mejor usar un fichero de propiedades para cada idioma.
2. **Utilizar guías de diseño**: Ribbon, Lolipop (Material Design de Google), etc.

# Accesibilidad.

Existen distintos tipos de ayudas técnicas para discapacitados: teclados / ratones adaptados, lectores de pantalla o dispositivos de reconocimiento de voz. Este tipo de hardware, conocido como tiflotecnología depende de que el software sea accesible para poder funcionar adecuadamente.

**Como desarrolladores**, debemos:

1. Meterle **Tooltips** a todo lo que veamos:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Permitir que el programa se pueda usar sin ratón (únicamente a golpe de tecla Tab y únicamente con teclado).
2. Agregar los componentes de un grupo en un panel. Por ejemplo, los radiobutton a un buttonGroup.
3. Siempre que una Label etiquete otro componente, utiliza el metodo setLabelFor.

# Formularios, formas y layouts.

1. Una buena separación visual la podemos conseguir dejando espacios en blanco que hagan de bordes. Si queremos destacar algún elemento, podemos usar un borde con línea

|  |  |
| --- | --- |
| Gráfico, Gráfico de barras  Descripción generada automáticamente | Igual altura indica similar función, pertenencia… |
| Simetría arriba y abajo, e izquierda y derecha. | Respetar los baselines |
|  |  |
| Los componentes están en su sitio.  Los labels no desplazan a los textboxes (**alineación a la derecha**). | Usar una anchura mínima para los botones. |
| El programa debe escalar bien a diferentes densidades de puntos y respetar las proporciones. Ejemplo donde no las respeta: | En la alineación de labels a la izquierda, debe haber suficiente espacio en blanco para no recargar la interfaz. |
| Uso de formas estables. Usar relaciones de aspecto conocidas y comunes. |  |

1. **La validación en formularios** nos sirve para cumplir el principio de usabilidad 4: minimizar la posibilidad de cometer errores.

Interfaz de usuario gráfica, Texto, Aplicación, Chat o mensaje de texto

Descripción generada automáticamente

Sin embargo, una opción más respetuosa con el usuario (principio 1, **adoptar el punto de vista del usuario**) es mostrar un mensaje de error. De esta manera, permitimos que el usuario pueda salir del campo de texto sin introducir un valor válido (que puede que no sepa introducir), y le informamos del error.