Instituto Politécnico Nacional
Escuela Superior de Cómputo
Ingeniería en Sistemas Computacionales
Desarrollo de aplicaciones móviles nativas


# Interfaces de usuario

Profesor: M. en C. Gabriel Hurtado Aviles

# Introducción

El diseño de interfaces de usuario (UI) es un campo fundamental en el desarrollo de aplicaciones y sistemas interactivos, ya que permite la interacción efectiva entre los usuarios y las máquinas. A lo largo de los años, han surgido diversas metodologías y herramientas para facilitar la creación de interfaces amigables, intuitivas y funcionales. En este contexto, el presente estudio se centra en analizar las principales formas de diseño de interfaces de usuario, con un enfoque particular en tres áreas clave que son esenciales para el desarrollo de interfaces en entornos modernos: el uso de asistentes de diseño, la implementación mediante XML (Extensible Markup Language) y el diseño directo a través de código.

# Introducción

El objetivo principal de esta presentación es proporcionar una visión general de cada una de estas aproximaciones, resaltando sus características, ventajas y limitaciones, con el fin de ofrecer un marco comparativo que sirva a desarrolladores y diseñadores en la toma de decisiones al momento de elegir la mejor opción para sus proyectos.

En los siguientes apartados, se abordarán los tres subtemas específicos: los asistentes de diseño de UI, el uso de XML como herramienta para la estructuración de interfaces, y la codificación manual, con el fin de profundizar en sus funcionalidades y en su impacto en el diseño y desarrollo de experiencias de usuario.

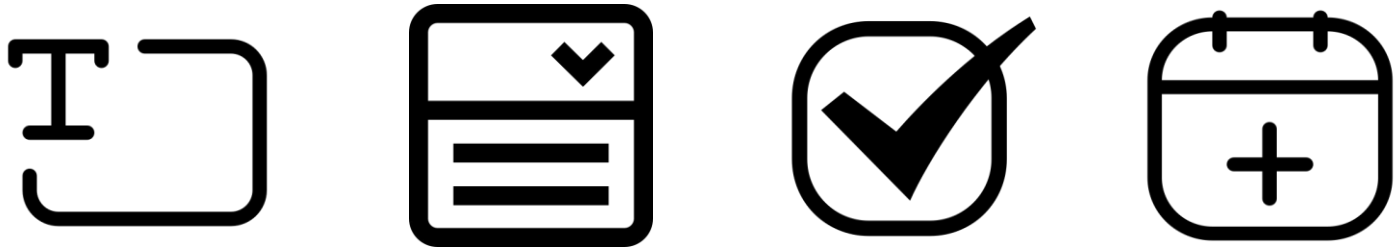
# Elementos de la interfaz de usuario

Una interfaz de usuario (UI) es el medio a través del cual un usuario interactúa con un dispositivo o aplicación. Su propósito es facilitar la comunicación entre el ser humano y la máquina, optimizando la experiencia del usuario al permitirle controlar y recibir información del sistema.

## Elementos de Entrada

Estos permiten a los usuarios ingresar información en el sistema. Algunos ejemplos incluyen:

- Campos de texto
- Listas desplegables
- Casillas de verificación
- Selectores de fecha



Elementos de Salida

Estos muestran resultados basados en las entradas del usuario. Ejemplos son:

- Alertas
- Mensajes de éxito

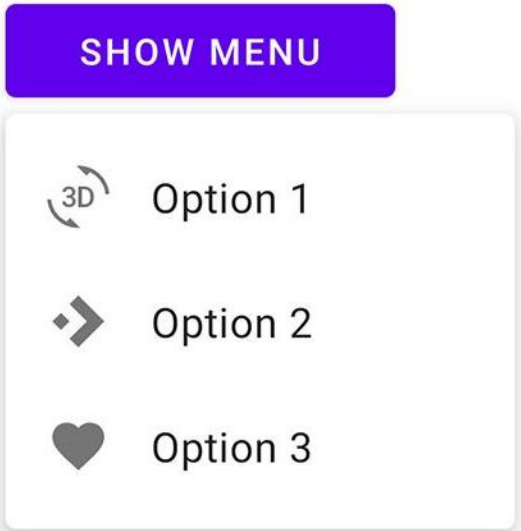
Elementos Auxiliares

Proporcionan información adicional y facilitan la navegación. Se dividen en:

- Elementos de navegación
- Elementos informativos
- Contenedores

## Controles

Estos son componentes interactivos que permiten al usuario realizar acciones, como botones, deslizadores y menús contextuales.



Views (Vistas)

Las vistas son los componentes visuales que los usuarios ven e interactúan.

- TextView: Muestra texto en la pantalla.
- EditText: Permite la entrada de texto.
- Button: Un botón que el usuario puede pulsar.
- ImageView: Muestra imágenes.
- ListView: Presenta una lista de elementos en forma vertical.
- Spinner: Un menú desplegable que permite seleccionar una opción.

ViewGroups

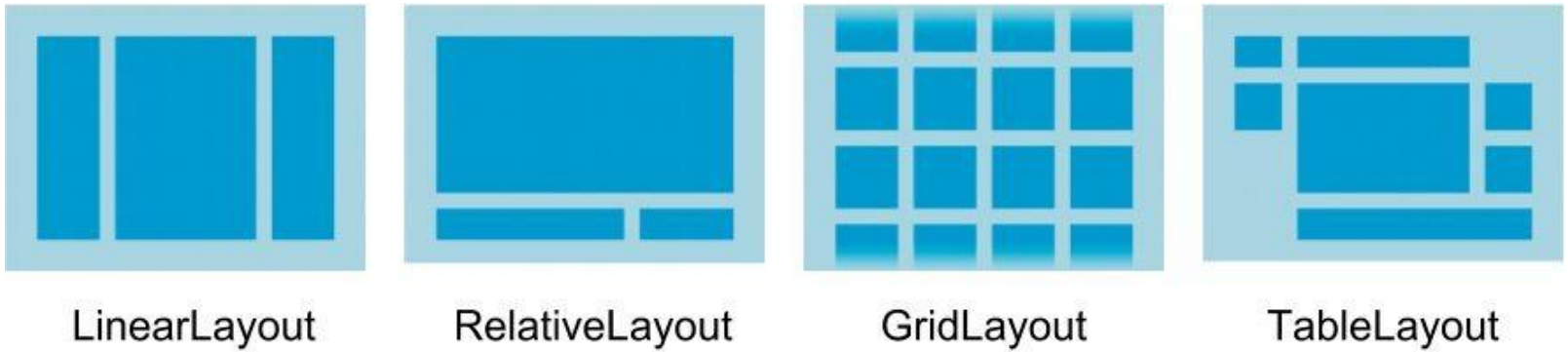
Estos son contenedores que agrupan varias vistas y organizan su disposición.

- LinearLayout: Organiza las vistas en una sola dirección, ya sea vertical u horizontal.
- RelativeLayout: Permite posicionar vistas en relación unas con otras.
- ConstraintLayout: Ofrece un diseño más flexible y adaptable, permitiendo relaciones complejas entre vistas.

## Layouts

Los layouts definen la estructura visual de la interfaz. Se pueden declarar en XML o programáticamente.

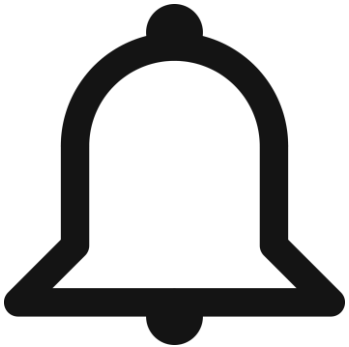
- ConstraintLayout: Permite un diseño adaptable y flexible.
- FrameLayout: Diseñado para contener un solo elemento.
- GridLayout: Organiza las vistas en una cuadrícula.



## Controles Auxiliares

Estos elementos ayudan a mejorar la experiencia del usuario, incluyendo:

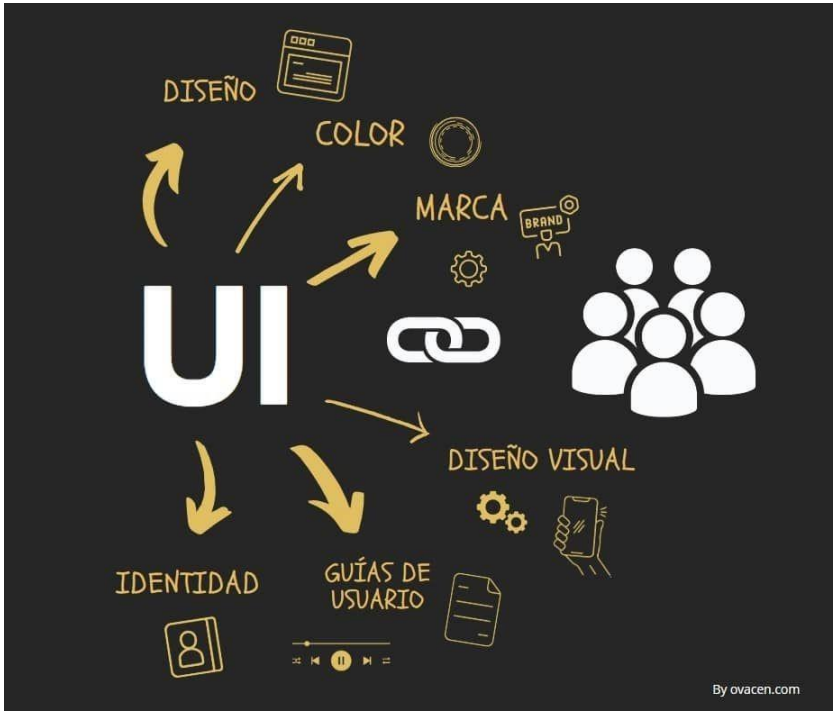
- Notificaciones: Mensajes que informan al usuario sobre eventos importantes.
- Iconos: Representaciones gráficas que ayudan a identificar acciones o funciones.



# Formas de diseño de la interfaz de usuario - Asistente

Las interfaces de usuario asistente (UI asistente) están diseñadas para interactuar de manera dinámica y proactiva con los usuarios.

Estas interfaces utilizan tecnologías como inteligencia artificial y procesamiento de lenguaje natural para ofrecer asistencia personalizada.



# Formas de diseño de la interfaz de usuario - Asistente

**Interactividad dinámica:** Responden en tiempo real a las solicitudes de los usuarios mediante diálogo o acciones automatizadas.

**Personalización:** Se adaptan a las preferencias y necesidades del usuario, aprendiendo de las interacciones previas.

**Soporte de lenguaje natural:** Pueden entender comandos hablados o escritos en lenguaje cotidiano, facilitando la interacción.

**Automatización de tareas:** Ejecutan tareas de manera automática, como programar citas, buscar información o controlar dispositivos.

**Facilidad de uso:** Orientadas hacia una experiencia intuitiva, eliminando barreras técnicas para el usuario final.

- Formas de diseño de la interfaz de usuario - Asistente

**Asistentes virtuales por voz:** Utiliza reconocimiento de voz para interactuar con el usuario. Ejemplos: Siri, Alexa, Google Assistant.

**Chatbots de texto:** Programas que mantienen conversaciones por escrito, resolviendo dudas o realizando tareas específicas. Ejemplos: Chat GPT, asistentes en sitios web.

**Asistentes integrados en aplicaciones:** Asistentes dentro de apps o plataformas que guían al usuario en el uso de herramientas específicas, como tutoriales interactivos o sistemas de soporte.

- Formas de diseño de la interfaz de usuario - XML

Se refieren a la manera en que se pueden definir y organizar interfaces gráficas en aplicaciones,a través de XML (eXtensible Markup Language), se puede estructurar de manera declarativa la disposición de elementos visuales sin necesidad de usar código imperativo.

## Linear Layout (Disposición Lineal)

- **Descripción**: Organiza los elementos uno detrás de otro, ya sea de forma vertical u horizontal.
- **Aplicación**: Común en Android. Por ejemplo, se pueden colocar botones o textos en una fila o columna.

## Relative Layout (Disposición Relativa)

- **Descripción**: Permite colocar los elementos en relación a otros elementos o al contenedor.
- **Aplicación**: En Android, este enfoque facilita el diseño de interfaces más complejas donde los elementos dependen entre sí (por ejemplo, un botón colocado debajo de otro).

## Constraint Layout (Disposición por Restricciones)

- **Descripción**: Se basa en restricciones que definen la posición y el tamaño de los elementos, lo que lo convierte en uno de los layouts más flexibles.
- **Aplicación**: Usado en Android, donde cada vista tiene restricciones en relación a otras vistas o a los bordes del contenedor.

## Grid Layout (Disposición en Cuadrícula)

- **Descripción**: Organiza los elementos en una cuadrícula, permitiendo posicionar elementos en filas y columnas.
- **Aplicación**: Común en Android para interfaces donde se necesita organizar elementos en una estructura de tabla o cuadrícula.

## Ventajas del uso de XML para interfaces

- **Separación de lógica y presentación**: La interfaz se define de manera independiente del código lógico.
- **Facilidad de personalización**: Modificar el diseño es más sencillo sin tocar el código fuente.
- **Reutilización**: Los archivos XML pueden ser reutilizados en múltiples componentes o pantallas.

# Formas de diseño de la interfaz de usuario - Código

## 1. Diseño basado en plantillas

Este enfoque utiliza **plantillas predefinidas** para crear la estructura de la interfaz. Las plantillas permiten ahorrar tiempo y ofrecen una coherencia visual. Las plantillas también son flexibles, lo que facilita personalizarlas para ajustarse a las necesidades del proyecto.

- **Ventajas:** Reduce el tiempo de desarrollo y proporciona consistencia visual.
- **Desventajas:** Puede limitar la creatividad y no siempre se adapta a proyectos específicos.

```html
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0/cs
</head>
<body>
  <div class="container">
    <header>
      <h1 class="text-center">Mi aplicación</h1>
    </header>
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
      <a class="navbar-brand" href="#">Inicio</a>
      <a class="nav-link" href="#">Contacto</a>
    </nav>
    <main>
      <div class="row">
        <div class="col-md-4">Contenido</div>
        <div class="col-md-8">Más contenido</div>
      </div>
    </main>
    <footer class="text-center">
      <p>© 2024 Mi aplicación</p>
    </footer>
  </div>
</body>
</html>
```

- Formas de diseño de la interfaz de usuario - Código

## 2. Diseño adaptable (Responsive)

El diseño adaptable ajusta la interfaz a diferentes tamaños de pantalla y dispositivos. Esto es fundamental en epocas recientes gracias a diferentes tipos de dispositivos cómo teléfonos inteligentes o tablets.

- **Ventajas:** Ofrece una experiencia de usuario óptima en cualquier dispositivo.
- **Desventajas:** Puede ser más complejo de implementar y probar.

- Formas de diseño de la interfaz de usuario - Código

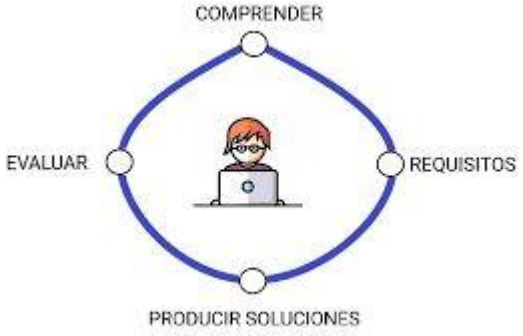
### 3. Diseño centrado en el usuario (User-Centered Design - UCD)

El diseño centrado en el usuario involucra al usuario desde el principio del proceso de diseño. Esto incluye **prototipado**, **pruebas de usuario** y **retroalimentación continua**.

- **Ventajas:** El diseño refleja mejor las necesidades y deseos del usuario.
- **Desventajas:** Puede ser un proceso más largo y costoso.

**Proceso de diseño UCD:**

1. Investigación de usuarios.
2. Prototipos rápidos.
3. Pruebas de usuario.
4. Iteración constante del diseño basado en la retroalimentación.



# Formas de diseño de la interfaz de usuario - Código

## 4. Diseño minimalista

El enfoque minimalista se centra en reducir al mínimo los elementos en la interfaz para evitar la sobrecarga de información. Utiliza mucho espacio en blanco y colores simples.

- **Ventajas:** Aumenta la claridad y la simplicidad de la interfaz.
- **Desventajas:** Si no se hace correctamente, puede resultar en una interfaz vacía o confusa

```html
<!DOCTYPE html>
<html>
<head>
  <style>
    body {
      font-family: sans-serif;
      margin: 0;
      padding: 20px;
      background-color: #f4f4f4;
    }
    .container {
      max-width: 600px;
      margin: auto;
      padding: 20px;
      background-color: white;
      text-align: center;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Interfaz Simple</h1>
    <p>Menos es más</p>
  </div>
</body>
</html>
```

- Formas de diseño de la interfaz de usuario - Código

## 5. Diseño esqueumofórico

Este diseño intenta imitar objetos del mundo real, utilizando sombras, texturas y efectos visuales que hacen que los elementos digitales se vean como objetos físicos.

- **Ventajas:** Ayuda a los usuarios a identificar fácilmente la función de un elemento.
- **Desventajas:** Puede parecer desactualizado en comparación con las tendencias actuales de diseño plano.

- Formas de diseño de la interfaz de usuario - Código

## 6. Diseño material

Introducido por Google, el diseño material utiliza elementos visuales como **sombras**, **transiciones** y **jerarquías claras** para crear interfaces que se sientan naturales.

- **Ventajas:** Mejora la experiencia visual e interactividad.
- **Desventajas:** Puede requerir más recursos gráficos.

```html
<!DOCTYPE html>
<html>
<head>
  <link href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/material
</head>
<body>
  <nav>
    <div class="nav-wrapper">
      <a href="#" class="brand-logo">Logo</a>
    </div>
  </nav>
  <div class="container">
    <h1>Material Design</h1>
    <p>Este es un ejemplo de interfaz con Materialize.</p>
  </div>
</body>
</html>
```

# Conclusión

Las formas de diseño de la interfaz de usuario han evolucionado para adaptarse a diversas necesidades de desarrollo, funcionalidad y experiencia del usuario.

En conclusión, las diferentes formas de diseño de interfaces de usuario ofrecen soluciones adaptadas a las necesidades específicas del proyecto y del equipo de desarrollo. Los asistentes simplifican y aceleran el proceso para diseños estándar, XML proporciona una estructura organizada y separación de componentes, y el uso directo de código otorga mayor control y personalización. La elección de la metodología dependerá del contexto del proyecto, las habilidades del equipo y los objetivos de la aplicación.