P9. Elaboración de interfaz gráfica y mapa de navegación cumpliendo con reglas de usabilidad y accesibilidad app móvil

¿Qué es una interfaz gráfica?

Es el conjunto de elementos visuales (botones, menús, íconos, pantallas) con los que el usuario interactúa dentro de una aplicación. Debe ser intuitiva, ordenada, y adaptada al tamaño de las pantallas móviles.

¿Qué es un mapa de navegación?

Es una representación visual, generalmente en forma de árbol jerárquico, que muestra cómo se organiza el recorrido del usuario entre las diferentes pantallas o secciones de una aplicación móvil.

Diferencia entre los tipos:

- Lineal
- Jerárquico
- Estructura web
- Estructura compuesta
  (Puedes usar una tabla comparativa o ejemplos gráficos)

Usabilidad y Accesibilidad

Usabilidad: se refiere a **qué tan fácil y eficiente** es para un usuario utilizar la aplicación:

- Rápido entendimiento
- Fluidez en las acciones
- Interacción intuitiva

Accesibilidad: asegura que personas con **discapacidades visuales, motoras, auditivas o cognitivas** puedan usar la aplicación correctamente:

- Textos legibles
- Contrastes adecuados
- Navegación por voz o teclado
- Alternativas visuales o sonoras

P10. Diseño responsivo

¿Qué significa?

El **diseño responsivo** (o "responsive design") es una técnica de desarrollo web que permite que una aplicación o sitio web **se adapte automáticamente al tamaño de pantalla del dispositivo** en el que se visualiza, ya sea:

- Teléfono móvil
- Tableta
- Computador de escritorio
- Televisor inteligente, etc.

Su objetivo principal es que el contenido sea **legible, funcional y atractivo**, sin importar el dispositivo que se utilice.

📲 ¿Por qué es vital en móviles?

1. **Uso mayoritario de móviles**: La mayoría de los usuarios acceden a sitios y apps desde teléfonos.
2. **Mejor experiencia de usuario**: Permite una navegación más cómoda, con menús adaptados al tacto, textos legibles y botones accesibles.
3. **Evita errores de uso**: El contenido no se desordena, no requiere hacer zoom manual ni desplazarse horizontalmente.
4. **Optimización del rendimiento**: Ayuda a cargar solo los elementos necesarios para el tamaño del dispositivo.
5. **Mejora el posicionamiento web (SEO)**: Google prioriza sitios con diseño responsivo en sus resultados de búsqueda.

📐 Ejemplos de adaptabilidad según resolución

Resolución	**Tipo de dispositivo**	**Cómo se adapta la interfaz**
320px - 480px	Teléfonos móviles pequeños	Menús desplegables, botones grandes, columnas apiladas
481px - 768px	Teléfonos grandes o tablets verticales	Columnas más anchas, tipografía ajustada
769px - 1024px	Tablets horizontales o pantallas medianas	Uso de 2 columnas, menús visibles sin desplegar
1025px en adelante	Computadores de escritorio	Diseño completo con múltiples columnas, elementos fijos

📌 *Un buen diseño responsivo detecta automáticamente el ancho de pantalla y ajusta los elementos del sitio sin necesidad de rediseñar cada versión.*

P11. Árbol jerárquico de navegación (propuesta práctica)

Explicación del diagrama de navegación de la aplicación móvil

El diagrama presentado representa el **flujo de navegación** de un usuario dentro de una aplicación móvil de pedidos o tienda en línea. A continuación, se describe cada parte y su función:

🔷 **Pantalla Principal**

Es el punto de entrada a la aplicación, desde donde el usuario puede:

- Ingresar al **Formulario de Registro**.
- Acceder al **Carrito de Compra**.
- Ir a la sección de **Ayuda**.

🔷 **Formulario de Registro**

Permite al usuario crear una cuenta o completar sus datos personales. Desde aquí se puede:

- Cancelar y volver a la pantalla principal.
- Acceder al **Formulario de Teléfonos**, que permite agregar un número de contacto.

🔷 **Formulario de Teléfonos**

Subpantalla del formulario de registro. Aquí el usuario añade su número telefónico y puede:

- Regresar al formulario anterior.
- Cancelar o finalizar el proceso de registro.

🔷 **Carrito de Compra**

Desde esta pantalla el usuario visualiza los productos seleccionados. Tiene las siguientes opciones:

- Cancelar y volver a la pantalla principal.
- Continuar al proceso de **Confirmar Pedido**.

🔷 **Confirmar Pedido**

Aquí se resumen los detalles de la compra. El usuario puede:

- Cancelar y regresar al carrito.
- Continuar hacia **Información de Envío**.

🔷 **Información de Envío**

Pantalla donde el usuario completa los datos para la entrega del pedido, como dirección y ciudad. Desde aquí:

- Se puede regresar al paso anterior (Confirmar Pedido o Carrito).

- O finalizar el proceso.

🔷 **Ayuda**

Sección informativa donde el usuario puede consultar dudas frecuentes o contactar con soporte. Desde aquí puede regresar fácilmente a la pantalla principal.

✅ **Resumen**

Este mapa de navegación evidencia un diseño **jerárquico** con rutas claras y bien definidas, que garantiza:

- **Facilidad de uso (usabilidad)**: el usuario puede navegar de forma lógica y predecible.
- **Accesibilidad**: hay rutas de retorno (“Cancelar” o “Listo”) en cada paso.
- **Experiencia amigable**: cada acción lleva a una nueva función o devuelve al usuario a una etapa previa sin confusión.
- Usa herramientas como Canva, Draw.io, Lucidchart o PowerPoint para crearlo.

P12. En la imagen se muestras es un mapa de navegación jerárquico lineal de una aplicación (probablemente un sistema de biblioteca o control de usuarios), donde las pantallas están organizadas desde un punto de entrada común hacia diferentes funcionalidades.

📌 Análisis del Mapa de Navegación:

Pantalla de Login (Inicio de sesión):

Punto de entrada inicial. Desde aquí el usuario accede al sistema.

Pantalla Home (Inicio principal):

Muestra un menú con varias opciones:

Multas

Mis artículos

Renovar

Reservar

Buscar

Configuración

Desde Home, el usuario puede ir directamente a:

🔶 Multas – Consulta de sanciones o pagos.

🔶 Mis Artículos – Listado de los elementos asignados al usuario.

🔶 Renovar – Extensión de tiempo de préstamo de artículos.

🔶 Reservar – Apartar artículos.

🔶 Buscar – Localizar artículos o contenido.

🔶 Configuración – Ajustes del sistema (como contraseña y notificaciones).

🔎 Tipo de estructura de navegación:

Esto representa claramente una:

✅ Navegación jerárquica con estructura lineal de segundo nivel

Jerárquica: Porque hay una pantalla principal desde la cual se accede a distintos módulos, organizados por importancia o funcionalidad.

Lineal (parcial): Aunque no es secuencial estricta, cada módulo funciona de forma aislada y sin saltos entre ellos, lo que le da un carácter lineal desde la pantalla principal hacia sus secciones.

✅ Ventajas de este tipo de navegación:

Es fácil de entender para los usuarios.

Permite organizar funcionalidades claramente.

Se adapta muy bien a aplicaciones móviles o kioscos interactivos.

Minimiza errores, ya que el usuario siempre parte del mismo punto central.

P16. Ejemplo.

1. Portada

- Nombre del aprendiz
- Ficha
- Nombre de la actividad (ejemplo: *Mapa de navegación y diseño de interfaz gráfica para aplicación móvil*)
- Fecha

---

2. Introducción

Breve descripción de lo que se busca con la actividad: organizar la información de la aplicación en un orden lógico, representarla en un **árbol jerárquico**, aplicar principios de diseño responsivo y buenas prácticas de usabilidad.

---

3. Objetivos

- Organizar la jerarquía del contenido de la aplicación móvil.
- Representar gráficamente el recorrido del usuario.
- Aplicar conceptos de diseño responsivo y navegación amigable.
- Elaborar plantillas iniciales de estilos para componentes.

---

4. Marco conceptual

Explica los conceptos clave que te piden:

- **Mapa de navegación:** qué es, para qué sirve.
- **Tipos de navegación:** lineal, jerárquico, estructura web, compuesta.
- **Diseño responsivo:** adaptación a diferentes tamaños de pantallas móviles.
- **Buenas prácticas de UI/UX:** claridad, consistencia, accesibilidad, jerarquía visual.

---

5. Requerimientos del cliente

Aquí describes qué información y funciones debe incluir la aplicación (por ejemplo, menú principal, secciones de productos, contacto, ayuda, etc.).

---

6. Orden lógico de la información

Redacta en forma de lista el contenido en el orden en que se mostrará al usuario. Ejemplo:

1. **Pantalla de inicio**
    - Logo / nombre de la aplicación
    - Botones principales (Ingresar / Registrarse / Continuar como invitado)
2. **Menú principal**
    - Perfil
    - Productos / Servicios
    - Carrito
    - Contacto
    - Ayuda
3. **Subsecciones**
    - Productos → Categorías (ejemplo: bebidas, comidas, postres)
    - Perfil → Datos personales, historial, configuración

---

7. Árbol jerárquico de navegación

Representa en forma de esquema o gráfico. Por ejemplo:

Inicio

├— Ingresar

├— Registrarse

└— Menú Principal

  ├— Perfil

  | ├— Datos Personales

  | ├— Historial

  | └— Configuración

  ├— Productos

  | ├— Bebidas

  | ├— Comidas

  | └— Postres

  ├— Carrito

  ├— Contacto

  └— Ayuda

Este esquema puede pasarse a un **mapa conceptual gráfico** (con diagramas tipo *árbol jerárquico*).

---

8. Plantillas y estilos de componentes

Puedes proponer **wireframes o bocetos** de las pantallas (aunque sea en PowerPoint, Figma, Canva o incluso dibujados). Incluye estilos básicos:

- Botones redondeados, colores principales de la app, tipografía, iconos.
- Ejemplo:
    - Botón principal → color azul, texto blanco, borde redondeado.
    - Fondo → degradado claro.
    - Texto → fuente sans-serif.

---

9. Conclusión

Menciona cómo la elaboración del mapa jerárquico y las plantillas facilitan la navegación y mejoran la experiencia de usuario.

P20. Definición de la Maquetación de la Interfaz Gráfica - Android

La actividad de **maquetación de la interfaz gráfica en el ecosistema Android** consiste en definir y organizar de manera estructurada todos los elementos visuales que conforman una aplicación móvil. Esta tarea se lleva a cabo principalmente mediante el uso del lenguaje **XML (eXtensible Markup Language)**, permitiendo diseñar pantallas funcionales y visualmente atractivas.

🖊️ Elementos considerados en esta maquetación:

- **Texto**: Definición del contenido textual, tipografía, tamaño (sp), color (hex) y alineación.
- **Formas**: Botones redondeados, tarjetas (CardView), contenedores (Layout), y sus respectivos bordes y sombreados.
- **Colores**: Selección de paleta cromática coherente con la identidad visual del proyecto, incluyendo colores primarios, secundarios y de acento (colors.xml).
- **Tamaños**: Uso de unidades como dp (density-independent pixels) para asegurar adaptabilidad a diferentes resoluciones de pantalla.
- **Imágenes**: Inclusión de íconos, logos y recursos gráficos (ImageView) almacenados en la carpeta res/drawable.
- **Animaciones**: Aplicación de transiciones, desvanecimientos, escalados u otras animaciones definidas en XML o programadas en Java/Kotlin.
- **Videos**: Inserción de recursos multimedia mediante componentes como VideoView o integraciones con servicios como YouTube API.

📐 Objetivo de esta actividad:

Garantizar una experiencia de usuario (UX) intuitiva, funcional y visualmente coherente, que facilite la interacción entre el usuario y la aplicación.

P21. Diseños y Vistas en Android: Los Layouts

En el desarrollo de aplicaciones móviles Android, los **layouts (diseños)** son estructuras que permiten **organizar y posicionar visualmente** los elementos de la interfaz gráfica de usuario (UI). Se definen mediante archivos XML que indican cómo deben disponerse las vistas (Views) dentro de un contenedor (ViewGroup).

🧱 ¿Qué es un Layout?

Un **Layout** es un **contenedor de vistas** que actúa como el marco donde se colocan botones, imágenes, textos, campos de entrada, etc.
Los layouts permiten definir la estructura visual y el comportamiento de los componentes en la pantalla.

P22. Principales Vistas de Interfaz en Android (Widgets)

A continuación, se describen algunos de los componentes más comunes que forman la interfaz gráfica en una aplicación Android:

Vista / Widget	Descripción
TextView	Etiqueta de solo lectura. Permite mostrar texto multilínea, cadenas formateadas, etc.
EditText	Caja de texto editable. El usuario puede ingresar texto; admite multilinea y texto flotante.
ListView	Lista vertical de elementos. Cada ítem es una vista y se corresponde con una fila de la lista.
Spinner	Lista desplegable. Permite seleccionar un valor entre varias opciones. Se visualiza como botón.
Button	Botón clásico que responde a eventos al ser pulsado.
CheckBox	Casilla que puede estar seleccionada o no. Representa opciones de tipo sí/no o múltiple.
RadioButton	Botones agrupados que permiten seleccionar una única opción entre varias disponibles.
SeekBar	Barra deslizante que permite seleccionar un valor dentro de un rango continuo (como volumen).

En la imagen muestra la **paleta de componentes comunes** en el **diseñador de interfaces gráficas de Android Studio** (modo gráfico o "Design View"). Aquí tienes una breve explicación de cada uno de los componentes que aparecen:

🧰 Componentes Comunes en la Paleta de Android Studio

Componente	Descripción
TextView	Muestra texto estático en pantalla. Ideal para títulos, etiquetas o mensajes.
Button	Botón que ejecuta una acción al pulsarlo. Es uno de los elementos más utilizados.

Componente	Descripción
ImageView	Permite mostrar imágenes (por ejemplo: íconos, fotos o recursos gráficos).
RecyclerView	Lista de elementos muy eficiente, ideal para mostrar grandes cantidades de datos. Reemplaza a ListView.
FragmentContainerView	Contenedor para alojar fragmentos dinámicamente. Útil en apps con navegación modular.
ScrollView	Permite desplazar verticalmente su contenido cuando excede el tamaño de la pantalla. Debe contener solo un elemento hijo.
Switch	Interruptor que representa un estado binario (encendido/apagado). Similar a un CheckBox visualmente más moderno.

P23. **En** La sigueinte diapositiva presenta una explicación sobre cómo se definen los **layouts** en Android mediante archivos **XML**, junto con un ejemplo de código que ilustra el uso de un LinearLayout vertical.

Resumen del contenido:

📌 **Texto explicativo (izquierda):**

- La manera habitual de usar *layouts* en Android es a través de un **fichero XML**, que se considera un recurso de la aplicación.
- Este archivo debe tener un **elemento raíz** (en este caso CoordinatorLayout), el cual puede contener múltiples layouts y vistas de forma anidada.
- Se muestra un ejemplo donde un TextView se coloca **encima** de un EditText dentro de un LinearLayout con orientación vertical.

🧠 Explicación línea por línea:

✅ Encabezado del archivo

<?xml version="1.0" encoding="utf-8"?>

- Define que el archivo es un XML y su codificación es UTF-8.

✅ CoordinatorLayout

<androidx.coordinatorlayout.widget.CoordinatorLayout

- Contenedor raíz del diseño. Permite organizar y coordinar vistas hijas (por ejemplo, barras de navegación, FloatingActionButton, etc.).

xmlns:android="http://schemas.android.com/apk/res/android"

- Espacio de nombres para usar los atributos estándar de Android.

xmlns:app="http://schemas.android.com/apk/res-auto"

xmlns:tools="http://schemas.android.com/tools"

- Permiten usar atributos personalizados y herramientas de diseño.

android:layout_width="match_parent"

android:layout_height="match_parent"

- El ancho y alto del layout ocuparán todo el espacio disponible de la pantalla.

android:fitsSystemWindows="true"

- Ajusta el contenido para evitar superposición con la barra de estado del sistema.

tools:context=".MainActivity">

- Indica la actividad asociada a este layout (solo para fines de desarrollo).

✅ LinearLayout

<LinearLayout

android:layout_width="match_parent"

android:layout_height="wrap_content"

android:orientation="vertical">

- Organiza los elementos hijos de arriba hacia abajo.
- wrap_content: la altura se ajusta al contenido.
- match_parent: el ancho ocupa todo el espacio disponible.

✅ TextView

<TextView

android:id="@+id/txtPantalla"

- Se le asigna un ID al TextView para referenciarlo desde el código.

android:layout_width="match_parent"

android:layout_height="150dp"

- Ancho completo, alto de 150dp (tamaño fijo en píxeles independientes de densidad).

android:layout_marginLeft="10dp"

android:layout_marginTop="40dp"

- Márgenes para separar el texto de los bordes izquierdo y superior.

P24. XML – Lenguaje de Marcado Extensible

✅ **¿Qué es XML?**

- **XML (Extensible Markup Language)** es un **metalenguaje** de marcado, es decir, un lenguaje diseñado para definir otros lenguajes de marcado personalizados.

✅ **Características clave:**

- 📌 **Metalenguaje**: Permite crear estructuras de datos personalizadas mediante etiquetas.
- 📌 **Lenguaje declarativo**: Se enfoca en el **contenido y estructura**, no en el comportamiento.
- 📌 **Estándar del W3C**: Fue desarrollado por el **World Wide Web Consortium (W3C)** y no por una empresa privada.
- 📌 **Multiplataforma**: Puede usarse en cualquier sistema operativo (Windows, Linux, macOS, etc.).
- 📌 **Simple y fácil de aprender**: Su sintaxis es sencilla y legible para humanos.
- 📌 **Flexible**: No tiene etiquetas predefinidas, tú defines las que necesites.
- 📌 **Extensible**: Puedes crear tus propias reglas y estructuras.
- 📌 **Optimizado para Internet**: Ideal para intercambiar datos entre sistemas a través de redes.
- 📌 **Libre y gratuito**: No requiere licencias de uso.
- 📌 **Facilita el desarrollo de herramientas**: Existen muchas bibliotecas y herramientas de procesamiento para XML en diversos lenguajes (Java, Python, C#, etc.).

✅ **¿Dónde se usa XML?**

- En el desarrollo web (por ejemplo, **layouts en Android**).
- Configuraciones de software (.xml en Android, Maven, etc.).
- Intercambio de datos entre aplicaciones (SOAP, RSS, SVG, etc.).
- Almacenamiento de datos estructurados.

P25. Uso de XML en el Desarrollo Web

✅ **¿Para qué se utiliza XML?**

- **Separación de datos y presentación**:
  XML **no define cómo se presenta la información**, sino que **almacena los datos**, permitiendo que sean reutilizados y presentados de distintas formas según el contexto o aplicación.
- **Interoperabilidad con HTML**:
  XML se **usa junto a HTML** en muchas aplicaciones.
    - XML **almacena o transporta los datos**.
    - HTML **se encarga de presentarlos o darles formato visual**.
- **Facilidad de actualización**:
  Dado que XML **guarda los datos en archivos separados**, no es necesario editar el archivo HTML cada vez que cambian los datos.
  Esto facilita el mantenimiento y actualización de las aplicaciones.

✅ **Uso con lenguajes de programación web**

- XML se puede **leer, procesar y manipular** desde lenguajes como:
    - **JavaScript** (por ejemplo, usando XMLHttpRequest o fetch)
    - **PHP**, **Python**, **Java**, entre otros.
- Gracias a esto, se puede **actualizar dinámicamente el contenido** de cualquier página web basada en HTML.

📄 **Sintaxis básica de inicio de un documento XML**

Todo documento XML comienza con la siguiente línea (opcional pero recomendable):

xml

CopiarEditar

<?xml version="1.0" encoding="UTF-8"?>

Explicación:

- version="1.0": indica la versión del estándar XML que se está utilizando (la más común es la 1.0).
- encoding="UTF-8": especifica la codificación de caracteres usada en el archivo (UTF-8 es la más utilizada y soporta caracteres internacionales).

P26. Reglas y buenas prácticas a cumplir

Los **documentos XML tienen una estructura jerárquica en forma de árbol**, donde:

- El **elemento raíz (root)** es el contenedor principal.

- Los **elementos hijos** son las ramas.
- Los **elementos sin hijos** son las hojas del árbol.

🎬 **Ejemplo de archivo XML: cine.xml**

xml

CopiarEditar

```
<?xml version="1.0" encoding="UTF-8"?>
<cine>
  <pelicula>
    <titulo>El Padrino</titulo>
    <director>Francis Ford Coppola</director>
    <anio>1972</anio>
    <genero>Drama</genero>
  </pelicula>

  <pelicula>
    <titulo>Interestelar</titulo>
    <director>Christopher Nolan</director>
    <anio>2014</anio>
    <genero>Ciencia Ficción</genero>
  </pelicula>

  <pelicula>
    <titulo>Parasite</titulo>
    <director>Bong Joon-ho</director>
    <anio>2019</anio>
    <genero>Thriller</genero>
  </pelicula>
</cine>
```

📚 **Explicación del archivo cine.xml**

Elemento	Descripción
<cine>	Elemento raíz. Contiene todas las películas.
<pelicula>	Elemento hijo de <cine>. Puede haber muchos (estructura repetible).
<titulo>, <director>, <anio>, <genero>	Hijos de cada <pelicula>. Son las "hojas", ya que no contienen otros elementos.

Este archivo **organiza una lista de películas**. Puede ser usado para:

- Mostrar contenido dinámico en una web.
- Almacenar información en una app.
- Ser leído por programas en JavaScript, Python, PHP, etc.

P27. Reglas de Sintaxis Básicas de XML

✅ **1. Debe tener un único elemento raíz (root)**

- Todo documento XML debe tener **un único elemento raíz** que contenga a todos los demás elementos.

```
<cine>
  <!-- otros elementos aquí -->
</cine>
```

✅ **2. El prólogo es opcional, pero debe ir en la primera línea si se usa**

- El **prólogo** especifica la versión de XML y la codificación de caracteres.
- Aunque es **opcional**, se recomienda incluirlo.

```
<?xml version="1.0" encoding="UTF-8"?>
```

- UTF-8 es la codificación estándar, adecuada para la mayoría de los idiomas.

✅ **3. Todos los elementos deben tener etiqueta de cierre**

- XML no permite etiquetas abiertas sin cierre como en HTML.

✅ Correcto:

```
<titulo>Matrix</titulo>
```

✅ **4. Las etiquetas son sensibles a mayúsculas y minúsculas**

- <Película> y <película> son **diferentes** para XML.

<pelicula> <!-- correcto -->

<Pelicula> <!-- es otro elemento distinto -->

✅ **5. Las etiquetas deben estar correctamente anidadas**

- Los elementos deben abrirse y cerrarse en orden correcto.

✅ Correcto:

<info>

<titulo>Matrix</titulo>

</info>

✅ **6. Los atributos deben ir entre comillas**

- Todos los valores de atributos deben ir entre **comillas dobles ("")** o simples ('').

✅ Correcto:

xml

CopiarEditar

<usuario nombre="Juan" edad="25" />

P28. Reglas y buenas prácticas a cumplir

Un elemento XML es cualquier cosa que esté incluida entre la etiqueta inicial y la final:

```
<director>George Miller</director>
```

Puede contener **texto**, **atributos**, **otros elementos** o una **mezcla de los anteriores**:

En la imagen es un ejemplo de archivo XML muy bien estructurado. A continuación explicaremos **línea por línea** del contenido del archivo selva.xml, ideal para ayudarte a entender cómo funciona este tipo de documentos.

📄 **Ejemplo: selva.xml**

<?xml version="1.0" encoding="UTF-8"?>

🔷 **Prólogo**: Declara la versión de XML y la codificación. Aquí se usa UTF-8, que es la más común.

<selva>

🔷 **Elemento raíz** del documento. Todo el contenido debe estar dentro de este elemento.

🐾 **Primer animal**

<animal>

<nombre>Werthers</nombre>

<tipo>Pantera</tipo>

<color>Negro</color>

<edad>12</edad>

</animal>

🔶 **<animal>**: Representa un animal dentro de la selva.

- <nombre>: "Werthers" – el nombre del animal.
- <tipo>: "Pantera" – especie o clase del animal.
- <color>: "Negro" – color del pelaje.
- <edad>: "12" – edad del animal en años.

 **Segundo animal**

<animal>

<nombre>Bun</nombre>

<tipo>León</tipo>

<color>Marrón</color>

<edad>15</edad>

</animal>

🔶 Otro nodo <animal> con:

- <nombre>: "Bun"
- <tipo>: "León"
- <color>: "Marrón"
- <edad>: "15"

</selva>

🔷 Cierre del elemento raíz.

Estructura de Árbol del XML

selva

├— animal

| ├— nombre: Werthers

| ├— tipo: Pantera

| ├— color: Negro

| └— edad: 12

└— animal

├— nombre: Bun

├— tipo: León

├— color: Marrón

└— edad: 15

Buenas prácticas cumplidas en tu archivo:

✓ Tiene un único elemento raíz: <selva>
✓ Todas las etiquetas están cerradas correctamente
✓ Uso correcto de mayúsculas y minúsculas
✓ Correcta anidación de etiquetas
✓ Sintaxis clara, limpia y válida

P38. Reglas y buenas prácticas a cumplir

Trabajo: Definición de la composición de una aplicación móvil en Android

Introducción

El desarrollo de aplicaciones móviles en el ecosistema Android requiere la integración de diversos elementos como texto, formas, colores, tamaños, imágenes, animaciones y videos, que en conjunto conforman una interfaz funcional, atractiva y acorde a los requerimientos del cliente. Para lograrlo, es fundamental conocer conceptos técnicos, de diseño gráfico y de maquetación, así como aplicar fundamentos de programación en XML y lenguajes web.

Objetivo

Definir la composición y estructura de los componentes gráficos y funcionales de una aplicación móvil en Android, considerando aspectos técnicos, estéticos y de usabilidad, con el fin de cumplir los requerimientos establecidos por el cliente.

Aspectos a tener en cuenta

Requisitos del cliente

- Identificación clara de las necesidades.
- Adaptación de los componentes al público objetivo.
- Inclusión de funcionalidades solicitadas.

Conocimientos técnicos necesarios

- Lenguaje XML: usado en la definición de layouts en Android.
- Layouts en Android: comprensión de LinearLayout, RelativeLayout, ConstraintLayout, entre otros.
- Responsividad: diseño adaptable a distintos tamaños y resoluciones de pantallas.
- Lenguajes web (HTML, CSS, JavaScript): apoyo en el diseño de prototipos y simulación de vistas.

Diseño y maquetación

- Maquetación gráfica: creación de prototipos visuales de la aplicación.
- Diseño gráfico: aplicación de colores, tipografía, íconos, imágenes y estilos visuales coherentes.
- Plantillas: estructuración de componentes repetibles para uniformidad en la app.
- Simuladores Android: herramientas como Android Studio y emuladores para la visualización de layouts y pruebas funcionales.

Conocimientos complementarios

- Marketing digital: apoyo en la definición de identidad visual y estrategias de posicionamiento de la app.
- Documentación: uso de herramientas para registrar maquetas y prototipos (Figma, Adobe XD, Balsamiq, entre otros).

Criterios de cumplimiento

- Identifica claramente los requerimientos del cliente.
- Conoce y aplica el concepto de maquetación gráfica.
- Domina el concepto de responsividad en interfaces móviles.
- Aplica fundamentos de programación en XML.
- Reconoce y utiliza layouts en Android.
- Implementa fundamentos de programación en HTML, CSS y JavaScript.
- Integra conceptos de diseño gráfico en los prototipos.
- Elabora maquetas y plantillas adecuadas para cada funcionalidad.

Conclusión

La creación de una aplicación móvil en Android no solo depende del conocimiento técnico en programación, sino también de la correcta aplicación de principios de diseño gráfico, responsividad y maquetación. Identificar los requerimientos del cliente, estructurar los layouts y utilizar simuladores adecuados garantiza que la aplicación sea funcional, atractiva y cumpla con los estándares de usabilidad y experiencia de usuario.

ANDROID STUDIO: COMO Crear una APP

https://www.youtube.com/watch?v=BQaxPwZWboA

Cómo crear un LOGIN SCREEN en Android Studio

https://www.youtube.com/watch?v=x66x9tltEX0

Completo por el momento no.

KOTLIN: Curso ANDROID desde CERO para PRINCIPIANTES

https://www.youtube.com/watch?v=ebQphhLpJG0