Dentro del **testing manual** existen varios tipos o enfoques que se usan según el objetivo, el momento del ciclo de vida del software y la técnica aplicada. Te hago un resumen de los principales:

🔹 Según el propósito de la prueba

1. **Testing funcional**
   - Verifica que el software haga lo que se espera según los requisitos.
   - Ejemplo: comprobar que al hacer login con usuario/contraseña válidos se accede al sistema.
2. **Testing no funcional**
   - Evalúa aspectos como rendimiento, usabilidad, seguridad, compatibilidad.
   - Ejemplo: medir tiempos de respuesta de la web en distintos navegadores.

🔹 Según el nivel en el ciclo de vida

1. **Testing de unidad (Unit Testing)** – normalmente se hace automático, pero en manual se puede revisar con pruebas muy puntuales en módulos pequeños.
2. **Testing de integración** – comprobar que dos o más módulos funcionan bien juntos.
3. **Testing de sistema** – validar la aplicación completa como un todo.
4. **Testing de aceptación (UAT)** – realizado por el cliente/usuario final para aprobar el producto.

🔹 Según el diseño de casos

1. **Caja negra**
   - Se centra en entradas y salidas sin importar el código.

- Ejemplo: probar un formulario con datos válidos e inválidos.

2. **Caja blanca (estructural)**
   - Se revisa el flujo interno, rutas de código, condiciones.
   - Aunque se hace más en automático, en manual también se revisan condiciones lógicas y caminos.
3. **Caja gris**
   - Mezcla de ambos enfoques: el tester conoce parcialmente la lógica interna.

🔹 Según la ejecución

1. **Exploratorio**
   - No hay guion rígido, el tester explora la aplicación buscando fallos y hay un check-list por ejemplo enfocarse en un módulo y buscar sólo fallos o errores graves
2. **Basado en casos de prueba**
   - Se siguen guiones predefinidos paso a paso.
3. **Ad-hoc**
   - Similar al exploratorio, pero más improvisado, sin documentación.

🔹 Tipos especiales

1. **Smoke Testing**
   - Comprobar funciones básicas para ver si la app “enciende”.
2. **Sanity Testing**
   - Verificar rápidamente que un bug corregido o una pequeña actualización no rompió nada.
3. **Regression Testing**

- Revisar que nuevas funciones no afectaron lo que ya funcionaba.

4. **Usability Testing**
    - Ver cómo los usuarios reales interactúan con el sistema.
5. **Compatibility Testing**
    - Probar en distintos dispositivos, navegadores o sistemas operativos.
6. **Accessibility Testing**
    - Validar que sea usable por personas con discapacidades (lectores de pantalla, navegación por teclado, etc.).
    -