

«Talento Tech»

Testing QA

Clase 13



Clase N°13 | Pruebas Mobile vs Desktop

Temario:

- Introducción al Mobile Testing
- Importancia de testear en dispositivos móviles
- Prácticas comunes en Mobile Testing
- Diferencias entre apps web, desktop y mobile
- ¿Qué cambia al adaptar una web a mobile?
- Estrategia Mobile First
- Sistemas operativos en entornos Desktop y Mobile
- Comparativa de pruebas: Desktop vs Mobile
- Herramientas para pruebas funcionales y visuales
 - Appium, BrowserStack, Firebase Test Lab, XCUITest, Espresso
 - Postman para testing de APIs en apps móviles
- Uso del Inspector del Navegador (F12) para simulación mobile
- Caso práctico: Talento Lab – Testing de una app mobile

Objetivos de la clase

En esta clase nos enfocaremos en las **pruebas de aplicaciones móviles**, sus diferencias frente al **testing en entornos desktop**, y los distintos enfoques que se deben considerar en función del dispositivo, sistema operativo y tipo de aplicación. Comprenderemos la importancia de validar la responsividad, la compatibilidad y el rendimiento en una amplia gama de entornos. Además, exploraremos casos prácticos, herramientas aplicables, buenas prácticas y una historia realista dentro del entorno de trabajo.

Mobile Testing

¿Qué es Mobile Testing?

El **Mobile Testing** es el proceso de probar aplicaciones que se ejecutan en dispositivos móviles como smartphones y tabletas. Incluye la validación de funcionalidades, interfaz de usuario, compatibilidad entre dispositivos, comportamiento ante interrupciones (como llamadas o notificaciones), consumo de batería y rendimiento.

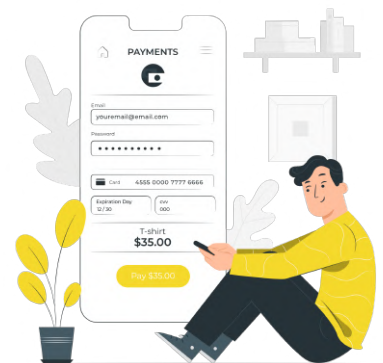


Importancia del Mobile Testing

- El uso de dispositivos móviles supera ampliamente al de computadoras de escritorio.
- Los usuarios esperan una experiencia fluida, rápida y sin errores.
- Cualquier fallo puede significar una pérdida de usuarios o mala reputación.
- Las tiendas de aplicaciones pueden rechazar o penalizar apps que no cumplan con los estándares.

Prácticas comunes en Mobile Testing

- Probar en dispositivos físicos y emuladores.
- Verificar compatibilidad con múltiples resoluciones.
- Validar comportamiento offline (sin conexión).
- Comprobar interrupciones (llamadas, notificaciones).
- Testear consumo de batería y uso de memoria.
- Verificar actualización y retrocompatibilidad.

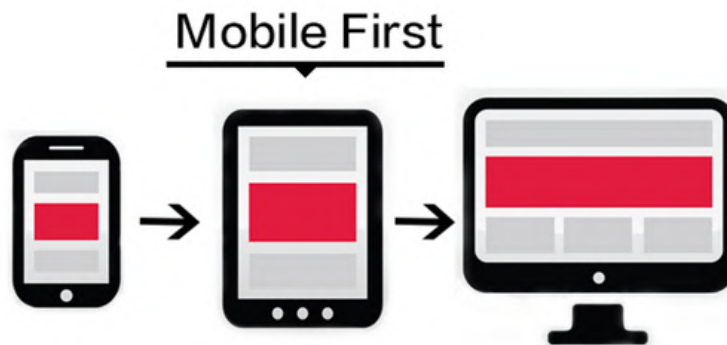


¿Qué se modifica al convertir una app web a mobile?

Cuando se transforma una aplicación web a mobile (generalmente a una app híbrida o nativa):

- Se elimina o rediseña contenido menos relevante para pantallas pequeñas.
- Se prioriza la interacción táctil.
- Se simplifican menús y navegación.
- Se adapta el layout para orientación vertical.
- Se optimiza el rendimiento en redes móviles lentas.

¿Qué es Mobile First?



Mobile First es una estrategia de diseño que prioriza la experiencia en dispositivos móviles desde el inicio del desarrollo. Primero se diseña para pantallas pequeñas, y luego se adapta progresivamente a pantallas más grandes.

Beneficios:

- Se enfoca en lo esencial.
- Reduce los tiempos de carga.
- Mejora la usabilidad.
- Obliga a simplificar y optimizar.

Ejemplo de aplicación con enfoque Mobile First:



La app de **Instagram** fue diseñada inicialmente pensando en usuarios móviles. Toda la navegación, el diseño vertical, los gestos y la experiencia general están optimizados para smartphones, y luego se adaptaron versiones para escritorio.

Sistemas Operativos en entornos Desktop y Mobile

Al realizar pruebas de software, es fundamental tener en cuenta el **sistema operativo (SO)** en el que se ejecuta la aplicación, ya que afecta directamente la compatibilidad, rendimiento y comportamiento de la app. Tanto en desktop como en mobile existen distintos entornos con sus particularidades.

En Desktop

1. Windows

- El sistema más utilizado en PC de escritorio.
- Requiere validar compatibilidad con versiones comunes (Windows 10, 11).
- Algunas apps pueden comportarse diferente según si son ejecutadas como administrador o con configuraciones de seguridad personalizadas.

2. macOS

- Presente en dispositivos Apple como MacBook y iMac.
- Puede tener diferencias en navegación de archivos, atajos de teclado o visualización de interfaces.
- Relevante para usuarios de diseño, marketing o desarrollo iOS.

3. Linux

- Utilizado por desarrolladores o servidores.
- Distribuciones como Ubuntu, Debian o Fedora presentan diferencias sutiles.
- Importante verificar el comportamiento en terminales, permisos de ejecución y dependencias.

En Mobile

1. Android

- Sistema operativo más extendido en el mundo.
- Alta fragmentación: múltiples versiones activas y fabricantes.
- Es clave testear en distintas marcas, resoluciones y versiones (Android 10, 11, 12, 13...).
- Permite instalación externa (APK), lo que requiere control extra de permisos.

2. iOS

- Exclusivo de Apple (iPhone y iPad).
- Menor fragmentación, pero más restricciones en seguridad y actualizaciones.
- Requiere validación con políticas de la App Store y funcionamiento con distintos tamaños de iPhone.
- Las versiones recientes cambian frecuentemente políticas de privacidad y UI.

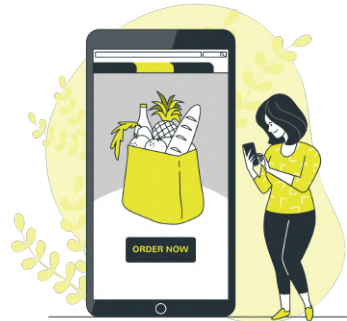
3. Otros SO (menos comunes)

- **HarmonyOS (Huawei), KaiOS, Tizen, FireOS, etc.**
En general, su uso es bajo, pero puede ser relevante si se apunta a mercados específicos (ej. India, China, dispositivos de bajo costo).
Se recomienda testear sólo si el producto lo requiere explícitamente.

Ejemplos de pruebas para Desktop vs Mobile

Aplicación Desktop (ejemplo: tienda online)

- **Navegación entre categorías:**
Verificar que los menús y enlaces a diferentes secciones (como Ropa, Electrónica, Ofertas) funcionen correctamente en todos los navegadores. Confirmar que las transiciones entre páginas sean rápidas y sin errores.
- **Comportamiento de formularios en distintos navegadores:**
Probar el formulario de registro o de pago en Chrome, Firefox y Edge. Validar campos obligatorios, formato de email, detección de errores en tiempo real y mensajes de validación.
- **Comprobación de enlaces rotos:**
Utilizar herramientas o pruebas manuales para asegurarse de que todos los botones, hipervínculos e imágenes redirijan correctamente y no devuelvan errores 404 o páginas vacías.
- **Compatibilidad entre navegadores:**
Confirmar que los elementos visuales (colores, botones, imágenes) se rendericen correctamente en cada navegador, manteniendo el diseño responsivo y la legibilidad.
- **Prueba de carritos, búsquedas, filtros:**
Simular un flujo de usuario agregando productos al carrito, utilizando filtros de búsqueda (por precio, categoría, popularidad), y validar que los resultados respondan correctamente al criterio elegido.



Aplicación Mobile (ejemplo: app bancaria)

- **Autenticación biométrica (huella, reconocimiento facial):**
Verificar que el sistema de autenticación biométrica funcione correctamente en dispositivos compatibles. Probar múltiples intentos, rechazos, y comportamiento cuando el usuario no tiene biometría configurada.
- **Uso de cámara y GPS:**
En funciones como escanear un cheque o localizar sucursales cercanas, testear si la app solicita los permisos adecuados, abre la cámara o GPS correctamente y actúa de forma esperada si el usuario niega los permisos.
- **Validación offline (modo avión):**
Probar funcionalidades en ausencia de conexión, como acceso al historial de transacciones almacenadas localmente o el comportamiento de la app ante errores.



de red. Verificar que los mensajes de error sean claros.

- **Prueba en diferentes tamaños de pantalla:**
Usar distintos dispositivos para comprobar que los textos, botones e imágenes se ajustan bien y no se solapan ni desbordan. Verificar que el diseño se mantenga usable en pantallas pequeñas o muy grandes.
- **Interacciones como swipe, tap, long press:**
Validar gestos táctiles: que un swipe despliegue el menú, que un tap abra una sección, y que un long press active opciones como eliminar o compartir. Asegurar que la respuesta sea consistente en todos los dispositivos.

Herramientas para testing Mobile y Desktop

Al realizar pruebas en entornos **mobile** y **desktop**, es importante elegir las herramientas adecuadas según el tipo de prueba que queramos ejecutar. A continuación, te presentamos algunas de las más utilizadas, explicando su propósito, alcance y ejemplos prácticos de uso:

- **Appium**

Appium es una herramienta de automatización de pruebas funcionales para **apps móviles nativas, híbridas y web**. Utiliza el lenguaje WebDriver, lo que permite escribir scripts en múltiples lenguajes como JavaScript, Python o Java.

¿Para qué se usa?

Para automatizar flujos de interacción reales en una app instalada, simulando toques, deslizamientos, validación de formularios, navegación, etc.

Ejemplo práctico: Simular el login en una app bancaria ingresando usuario, contraseña, y verificando si aparece la pantalla de bienvenida.

- **BrowserStack / SauceLabs**

Estas plataformas permiten hacer pruebas **cross-device y cross-browser en la nube**, utilizando dispositivos reales y navegadores reales sin necesidad de tenerlos físicamente.

¿Para qué se usan?

Para validar cómo se comporta una web o app en distintos modelos de celulares, resoluciones, sistemas operativos y navegadores.

Ejemplo práctico: Ejecutar tu sitio en un iPhone 13 con Safari y un Galaxy S22 con Chrome para verificar que el layout y la funcionalidad sean correctos en ambos casos.

- **Firebase Test Lab (Android)**

Servicio de Google que permite subir una app (APK o AAB) y ejecutarla automáticamente en una gran variedad de dispositivos Android físicos y virtuales.

¿Para qué se usa?

Para realizar pruebas de UI sin escribir código o ejecutar scripts de pruebas instrumentadas (Espresso, UI Automator).

Ejemplo práctico: Subir tu APK al Test Lab y recibir un informe de compatibilidad, capturas de pantalla, videos de ejecución y errores detectados.

- **XCUITest (iOS) y Espresso (Android)**

Son frameworks de automatización **nativos** diseñados específicamente para iOS y Android, respectivamente.

¿Para qué se usan?

Para realizar pruebas funcionales integradas con el código de la app, ideales para pipelines de CI/CD y pruebas en profundidad en apps productivas.

Ejemplo práctico: Verificar que un botón de “Confirmar compra” funcione correctamente en múltiples versiones de iOS usando XCUITest, o validar que una lista cargue correctamente en Android usando Espresso.

¿Se puede usar Postman para pruebas mobile?

Sí, pero con ciertas limitaciones.

Postman es una excelente herramienta para probar las **APIs que utiliza una app móvil**. Como muchas apps móviles consumen servicios backend vía HTTP, Postman permite simular esas solicitudes y validar:

- Que los endpoints respondan correctamente.
- Que los datos JSON tengan la estructura esperada.
Que las respuestas lleguen con los códigos de estado adecuados (200, 201, 400, etc.).

Limitación:


Postman **no prueba la interfaz gráfica ni las interacciones físicas del usuario** (como toques, gestos o uso de sensores). Para eso, necesitás herramientas como Appium o Espresso.


Ejemplo práctico: En una app de compras, podés usar Postman para verificar si el endpoint de creación de pedido devuelve correctamente el ID del pedido y un estado “confirmado”.

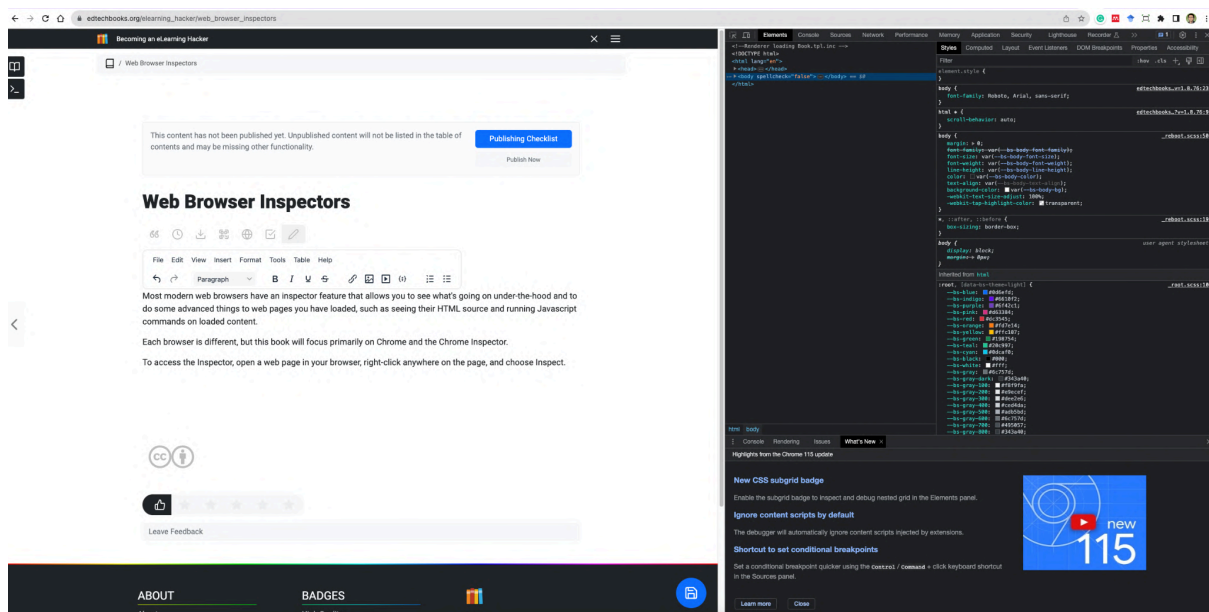
Uso del inspector del navegador (F12)

La mayoría de los navegadores como Chrome o Edge permiten simular un entorno móvil desde el **inspector (Developer Tools)** con F12.

Pasos:

1. Presionar F12 (o clic derecho > Inspeccionar).
2. Ir al icono de "Toggle device toolbar" ().
3. Elegir entre modelos como iPhone, Pixel, Galaxy, iPad.
4. Observar cómo se adapta el diseño responsivo.

 Ideal para detectar fallos visuales sin tener un dispositivo real a mano.



Material Complementario

- [Guía oficial de Appium \(documentación\)](#)
- [BrowserStack para testing cross-browser y mobile](#)
- [Firebase Test Lab \(Google Cloud\)](#)
- [Documentación de XCUITest - Apple Developer](#)
- [Documentación oficial de Espresso - Android Developers](#)

¡Trabajando en Talento Lab!

Silvia y Matías te esperan en la sala de reuniones con una nueva tarea. El equipo de Talento Lab está por lanzar una nueva funcionalidad de su plataforma para postulantes: una app móvil que complementará el sitio web existente. Tu responsabilidad será asegurar que esa app funcione correctamente en distintos dispositivos y sistemas operativos.



Silvia te explica: “Como ya tenemos la versión web funcionando, vamos a lanzar una versión mobile. Pero no podemos asumir que todo funcionará igual. Quiero que enfoques las pruebas en la responsividad, la compatibilidad con Android e iOS, y las funcionalidades críticas como la carga de CV y el inicio de sesión.”



Matías te comparte una lista de dispositivos disponibles para pruebas, junto con una serie de scripts para ejecutar en Appium. “Acordate de validar cómo se comporta la app si se recibe una llamada durante el uso, si pierde la conexión o si se cambia la orientación del dispositivo.”

También te pide que explores la vista mobile del sitio actual usando el inspector del navegador y anotes cualquier diferencia visual respecto a lo que se espera en la app. Tu trabajo será clave para garantizar que los usuarios tengan una experiencia fluida, sin importar desde qué dispositivo accedan a la plataforma.

Silvia concluye: “Recordá documentar cualquier hallazgo con capturas de pantalla y recomendaciones. La idea es lanzar una experiencia mobile sin sorpresas.”

Ejercicio: Responsividad y compatibilidad Desktop vs Mobile en Talento Lab

Instrucciones

1. Abre <https://talentolab-test.netlify.app/> en tres entornos:
 - **Desktop** (p. ej. Chrome en 1920×1080)
 - **Tablet** (p. ej. Safari o Chrome en 768×1024)
 - **Móvil** (p. ej. Chrome o Safari en 390×844)
2. Para cada uno de los siguientes módulos/funcionalidades, realiza las pruebas descritas en la tabla.
3. Toma capturas de pantalla de cada dispositivo y anótalas junto con tus observaciones.
4. Completa un documento (Excel o Word) con la siguiente estructura:



Responsividad y compatibilidad...

Entrega

- **Documento** (Excel o Word) con la tabla completa, una fila por módulo y captura(s) de pantalla para cada dispositivo.
- **Capturas** renombradas como “Desktop_Navegación.png”, “Tablet_Servicios.png”, “Mobile_Contacto.png”, etc.
- **Observaciones** breves junto a cada captura (p. ej. “En móvil, el campo de email queda bajo el teclado, no es visible”).

Con este ejercicio asegurarás que todas las secciones críticas de Talento Lab ofrecen una experiencia consistente y usable, tanto en desktop como en tablet y móvil.

Próximos Pasos

En la próxima clase vamos a comenzar a trabajar con **Jira**, una herramienta clave para la gestión de proyectos ágiles y el seguimiento de tareas desde el rol de QA.

¿Qué aprenderás en la Clase 14?

- Cómo crear tu cuenta gratuita en Jira y entender su interfaz.
- Qué son los **tableros**, el **backlog** y cómo se estructuran los **proyectos** en Scrum.
- Diferencias entre los tipos de tickets: **épicas**, **historias**, **tareas**, **bugs** y **sub-tareas**.
- Cómo redactar una descripción efectiva de errores.
- Qué son los **Story Points**, cómo hacer una buena estimación con técnicas como **Planning Poker**.
- Cómo asignar tareas, agregar etiquetas, y adjuntar evidencias como capturas de pantalla.

Esta clase marca el inicio del módulo donde vas a ver cómo **planificar, organizar, estimar y documentar tu trabajo como QA**, además de familiarizarte con el entorno de trabajo real en el que te vas a mover en proyectos ágiles.

¡Todo lo aprendido sobre testing en escritorio, mobile y APIs va a empezar a conectarse en un flujo real de trabajo!



Buenos Aires
aprende
Agencia de Habilidades para el Futuro

BA Buenos
Aires
Ciudad