

**MP\_0487. Entornos de desarrollo**

**UF3. Diseño y realización de pruebas**

**3.1. Tipos de pruebas**

# Índice

---

☰	Objetivos	3
☰	Entornos de trabajo	4
☰	El proceso de pruebas	6
☰	Validación y verificación	7
☰	Pruebas estáticas	8
☰	Pruebas dinámicas	9
☰	Pruebas funcionales	10
☰	Pruebas estructurales	12
☰	Pruebas de integración	14
☰	Pruebas de usabilidad y accesibilidad	16
☰	Pruebas de caja negra (detección de errores)	18
☰	Pruebas de seguridad	20
☰	Pruebas de rendimiento	21
☰	Ámbitos de aplicación de los distintos tipos de pruebas	23
☰	Planificación	27
☰	Estructura de un plan de pruebas	29
☰	Resumen	30

# Objetivos

---

En esta lección perseguimos los siguientes objetivos:

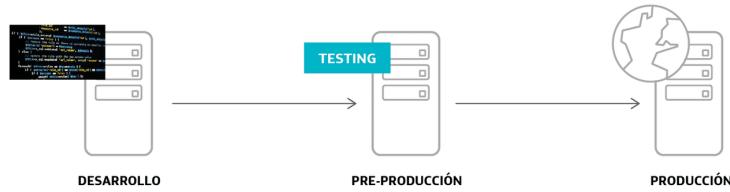
- 1 Identificar los tipos de pruebas a realizar durante el desarrollo de una aplicación.
  - 2 Analizar los puntos de prueba de un programa.
  - 3 Diseñar un plan de pruebas de una aplicación.
- 

¡Ánimo y adelante!

# Entornos de trabajo

Se denomina entorno de trabajo al conjunto de estados por los que pasa el proceso de desarrollo de una aplicación, así como las características de software, hardware y arquitecturas que la rodean a lo largo de su ciclo de desarrollo.

Estas circunstancias o entornos van a ser diferentes, dependiendo de la complejidad del desarrollo, pero en general se definen los siguientes entornos:



## Entorno de desarrollo

Es el entorno inicial, donde la aplicación es desarrollada por el grupo de programadores. Dicho entorno está enfocado al desarrollador, estableciéndose las configuraciones y medidas de seguridad adecuadas que permitan facilitar y agilizar dicho proceso de desarrollo. Generalmente el desarrollador tiene permisos de administración sobre el equipo en el que trabaja.

La infraestructura utilizada a nivel de software y hardware, en general, no va a coincidir con la final prevista para el lanzamiento de la aplicación, siendo de características más bajas con el objeto de reducir los costes de infraestructura.

## **Entorno de pre-producción**

Este es el entorno en donde se realizarán las pruebas oportunas para garantizar que el software se comportará de forma adecuada antes de pasar al entorno de producción. Por lo tanto, es requisito indispensable que este entorno se asemeje en la medida de lo posible al entorno de producción a nivel de arquitectura, niveles de seguridad, hardware, software etc.

El acceso a la aplicación estará permitida solo para el grupo de testeadores y con datos que, si no son los reales, al menos deberán ser muy semejantes.

## **Entorno de producción**

Se define el entorno de producción como aquel en el que la aplicación opera en su ubicación final, con los recursos de software y hardware con la que fue diseñada y con usuarios y datos reales, donde cualquier fallo o error puede llevar asociadas grandes pérdidas a la organización y, por tanto, la versión que se libera en este entorno ha debido ser previamente comprobada. Todo el proceso de subida a producción desde un estado previo debe ser gestionado y planificado de manera conveniente, estableciéndose un plan de contingencia que puede incluir un método de vuelta a atrás hasta el punto de partida (en estas circunstancias un *back-up* de todo el sistema implicado es imprescindible).

# El proceso de pruebas

---

El proceso de pruebas es una fase del denominado ciclo de vida del desarrollo de software, que consiste en demostrar que la aplicación funciona de la forma en la que había sido diseñada.

Es decir, se puede definir como un proceso que pretende validar y verificar que una aplicación:

## **Cumple los requisitos**

Cumple los requisitos del cliente y de tipo técnico para el cuál fue desarrollada. Lo que implica que deberán hacerse pruebas para cada uno de los requerimientos con los que se diseñó la aplicación.

## **Funciona de forma adecuada, tal y como se esperaba**

Por lo que en un proceso de pruebas se tratará de buscar y eliminar los posibles fallos de la aplicación.

# Validación y verificación

Según la definición aportada para un proceso de pruebas, parece entenderse que validación es sinónimo de verificación.

Sin embargo son términos con significados diferentes, siendo su significado:

## Validación

Este proceso trata de responder a la pregunta: ¿se está construyendo el sistema adecuado?

## Verificación

Este proceso trata de responder a la pregunta: ¿se está construyendo bien el sistema?

Es decir, la **validación** se refiere a la comprobación de que el sistema cumplirá con las necesidades reales del cliente, mientras que la verificación se refiere a si el sistema está bien diseñado y sin errores de funcionamiento.

La **verificación** tratará de determinar si el software tiene la calidad deseada, pero no va a asegurar que el sistema sea útil para el cliente. Puede suceder que un software verifique que funciona de forma adecuada y sin embargo no es válido por una toma errónea de requerimientos.

## Pruebas estáticas

---



Se corresponden con un tipo de pruebas en las que **no es necesario ejecutar la aplicación** para llevarlas a cabo.

En este tipo de pruebas se revisan e inspeccionan los requisitos, los prototipos de diseño, el código fuente de la aplicación o el diseño de pruebas de la aplicación en busca de posibles errores con el objetivo de eliminar defectos en las fases tempranas del desarrollo del software.

Las revisiones podrán realizarse de forma manual o bien por medio de herramientas automatizadas específicas.

En general, cualquier documentación puede ser revisada, como por ejemplo:

- Especificaciones de requisitos.
- Documentos de diseño.
- Códigos fuente.
- Planes de prueba.
- Casos de prueba.
- *Scripts* de prueba.
- Documentos de ayuda de usuario.
- Contenido de la página web.

## Pruebas dinámicas

---

Debido a la necesidad de ejecutar la aplicación, este tipo de pruebas no podrán realizarse en las fases más tempranas del desarrollo, sino que **deberán realizarse una vez se tenga disponible una versión ejecutable o prototipo de la aplicación.**

Se corresponden con un tipo de pruebas en las que para ser realizadas **es necesario ejecutar el código.**

El objetivo general de las pruebas es confirmar que el software se comporta en concordancia con los requerimientos.

Con estas pruebas se comprueba el comportamiento funcional del software, de la memoria, el uso de la CPU y el rendimiento general del sistema. Es decir, en este tipo de pruebas se realizarán tanto pruebas funcionales como no funcionales, tal y como se verá en secciones posteriores.

# Pruebas funcionales

En este tipo de pruebas el testeador no tiene por qué conocer cómo es la lógica interna del sistema.

Las pruebas **funcionales** constituyen un tipo de pruebas de software mediante las cuales se comprueba que se cumplen los requisitos de funcionalidad y las especificaciones de la aplicación.

Las funciones o características son comprobadas generando unos **datos de entrada** en la aplicación y analizando los resultados para poder comprobar si son tal y como se esperaban.

El objetivo de las pruebas funcionales, por tanto, es probar y validar que el software hace justo lo que en sus especificaciones se dice que hace.

En general, una prueba funcional típica se compone de las siguientes fases:

1

**Identificación de las funciones** que el software debe realizar.

2

**Generación de los datos de entrada** en base a las especificaciones de las funciones.

3

**Determinación de los elementos de salida** en base a las especificaciones de la función.

4

Ejecución de la prueba en base a los parámetros establecidos.

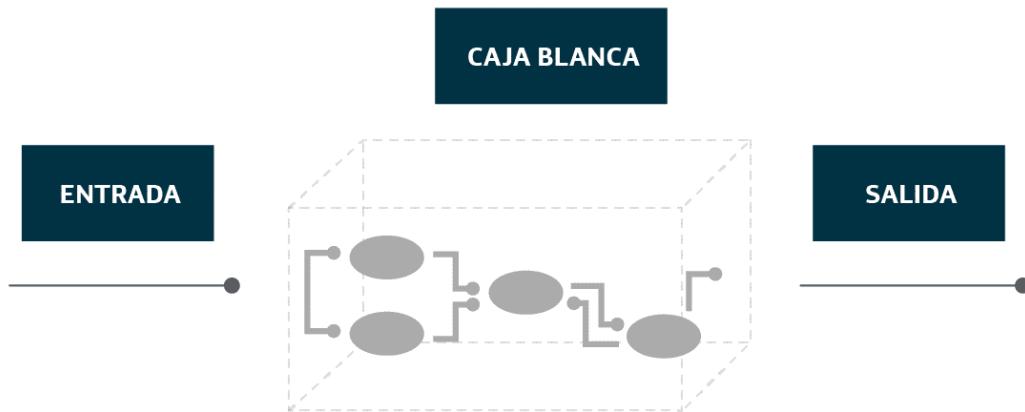
5

Comparación de los resultados obtenidos frente a los esperados.

- i** En contraposición a las pruebas funcionales se incluyen las denominadas pruebas no funcionales, que no están referidas a las funciones, sino a otros aspectos de la aplicación, como pueden ser las de rendimiento, carga, usabilidad, seguridad, etc.

# Pruebas estructurales

Las pruebas estructurales son aquellas en las que la estructura, el diseño y la implementación del sistema son conocidos por la persona que realiza la prueba. Por este motivo, a este tipo de pruebas también se les denomina como de '**caja blanca**' o 'caja de cristal'.



Para testear la aplicación se elige una entrada. A partir de esta se examina la aplicación por los distintos posibles caminos, obtenidos a partir del código, por los que se puede ejecutar la aplicación, revisando las salidas que se producen.

Durante las pruebas estructurales el testeador se centra en **determinar cómo trabaja el software**. Por ejemplo, tratará de saber cómo se comporta un bucle dentro del código. Para ello procederá a realizar diferentes casos de pruebas para cada uno de los distintos valores del bucle.

En general, las pruebas de caja blanca son aplicables a los siguientes ámbitos de pruebas:

### **Pruebas unitarias**

Se testean las posibles rutas dentro de una unidad.

### **Pruebas de integración**

Se testean los posibles caminos entre las unidades.

### **Pruebas de sistema**

Se testean diferentes caminos entre los subsistemas.

---

A continuación se mencionan las principales ventajas y desventajas de las pruebas estructurales o de “caja blanca”:

#### **VENTAJAS**

##### **Inicio en fases tempranas**

Ya que pueden realizarse desde el mismo momento en el que el código está implementado, sin necesidad de haber implementado, por ejemplo, la interfaz gráfica final de la aplicación.

##### **Pruebas completas**

Puesto que es posible verificar todos y cada uno de los posibles caminos.

#### **INCONVENIENTES**

##### **Precisa perfiles cualificados**

Con profundos conocimientos de programación y ejecución, puesto que las pruebas están basadas en la interpretación del código fuente y pueden llegar a ser muy complejas.

##### **Modificaciones continuas del script de pruebas**

En situaciones en las que el código de la aplicación cambia continuamente.

# Pruebas de integración

Son pruebas que se realizan después de las pruebas unitarias y antes de las pruebas de sistema. Se corresponden con un nivel de pruebas en donde las unidades independientes de un sistema son combinadas y comprobadas cuando trabajan en grupo.

El objetivo de las pruebas es detectar los errores derivados de las interacciones entre las unidades que se integran. De modo general se revisará que:

- Cada elemento interactúa de forma correcta con sus interfaces.
- Que de forma individual satisface la funcionalidad que le ha sido requerida.
- Que la funcionalidad del conjunto se realiza de forma correcta.



Para que las pruebas de integración se realicen de forma correcta deberán ser tenidos en cuenta los siguientes aspectos:

1

Que defina el detalle de las interacciones entre cada unidad.

2

Probar por separado cada unidad antes de ser integrado al conjunto.

3

En el supuesto de que se realicen pruebas repetitivas de regresión según se van incorporando nuevos componentes.

# Pruebas de usabilidad y accesibilidad

---

Este tipo de pruebas tienen que ver con la facilidad de uso de la aplicación, así como de la accesibilidad (contraste de color, accesible por teclado...).

## Usabilidad

La usabilidad de una aplicación puede entenderse como una medida en la que esta puede ser utilizada por los usuarios de forma clara, sencilla, efectiva y satisfactoria.

Para las pruebas de usabilidad se aconseja al menos la valoración de un mínimo de 5 usuarios. Podrán tanto ser expertos en temas de usabilidad como usuarios finales de distintos perfiles. Ya sean de un tipo o de otro, se les plantea una serie de preguntas a las que deberán contestar durante la pruebas.

---

El concepto de usabilidad está ligado a la experiencia que tiene un usuario a la hora de navegar por la aplicación web y, por tanto, la forma de implementar las pruebas será por medio de uno o varios usuarios que acceden y valoran su experiencia.

## Accesibilidad

Según la W3C, se habla de que un sitio web es accesible cuando “**el acceso se realiza de forma universal, independientemente del tipo de hardware, software, infraestructura de red, idioma, cultura, localización geográfica y capacidades de los usuarios**”.

Cada una de las categorías principales de discapacidad requiere un determinado tipo de adaptación. Se trata de realizar aplicaciones adaptadas para que accedan usuarios con discapacidad visual, de audición, de motricidad, cognitiva...

# Pruebas de caja negra (detección de errores)

Aunque por lo general las pruebas de **caja negra** suelen ser de tipo funcional, también se incluyen las no funcionales, como por ejemplo una prueba de rendimiento.

Las pruebas de caja negra o pruebas de “comportamiento”, son un método de prueba del software en el que la estructura interna del elemento a probar no es conocida por la persona que realiza el test. Es decir, a la vista del testeador el elemento se comporta como una caja negra.



En este tipo de pruebas se trata de detectar:

- Funciones incorrectas o que no existen.
- Errores en la interfaz.
- Errores en el acceso a los datos.
- Errores en el comportamiento.
- Errores en el inicio o terminación de la aplicación.

---

A continuación se mencionan las principales ventajas y desventajas de las pruebas de caja negra.

#### VENTAJAS

- **Se facilita la detección de no conformidades con las especificaciones**, puesto que se realizan desde el punto de vista del usuario.
- **No se requieren perfiles cualificados** para la realización de las pruebas, puesto que no se necesitan conocimientos de programación o de la implementación del software.
- **Perspectiva objetiva de la prueba**, puesto que se realiza por personal ajeno a los programadores.
- **Desarrollo temprano de los casos de pruebas**, puesto que pueden ser elaborados en cuanto se definen las especificaciones.

#### INCONVENIENTES

- **Algunas rutas no son probadas**, puesto que solo son definidas unas entradas determinadas.
- **Dificultad de diseñar las pruebas** en aquellas situaciones en las que las especificaciones no han sido bien definidas.

## Pruebas de seguridad

---

El objetivo de las pruebas de seguridad es asegurar que el **software estará libre de las amenazas** externas o internas procedentes de las personas y los programas maliciosos.

Este es un tipo de pruebas de software que es realizado por personal técnico cualificado especializado en dichos tipos de pruebas.

A la hora de realizar este tipo de pruebas se ha de tener en cuenta que existe un número ilimitado de formas con las que se puede atacar a una aplicación. Por lo tanto, nunca podrá asegurarse que una aplicación está libre de ataque. Aun así, se recomienda incluir las pruebas de seguridad como parte del proceso de desarrollo de software.

De modo general, en este tipo de pruebas se pretende comprobar:

- Si existe una autenticación adecuada.
- Si el control de acceso es adecuado.
- Si el software es capaz de mantener la confidencialidad de los datos.
- La disponibilidad del software en caso de un ataque por parte de *hackers* y programas maliciosos.

# Pruebas de rendimiento

En la mayoría de los sistemas de software que se desarrollan es imposible, a priori, determinar cuáles son los valores máximos que va a poder soportar la aplicación o cómo se va a comportar ante situaciones de carga muy elevadas.

En estos casos es necesario hacer las denominadas pruebas de comportamiento de la aplicación, que serán realizadas sobre los entornos de preproducción lo más cercanos posibles al de producción por herramientas adecuadas, que serán capaces de reproducir las condiciones de prueba requeridas, como pueden ser el número de usuarios concurrentes que acceden a la aplicación.

En las pruebas de rendimiento se somete a la aplicación a un conjunto de solicitudes simultáneas determinadas (por ejemplo se probará la aplicación con 50, 100 y 150 usuarios concurrentes) y se analizará su comportamiento para cada una de ellas.

Los principales parámetros obtenidos de una prueba de rendimiento son:

## Tiempos de respuesta

Es la métrica más solicitada en las pruebas de rendimiento. Afecta de forma directa a la experiencia con la aplicación del usuario final.

## El uso de recursos

Suelen ser de tipo narrativo, como por ejemplo “durante las pruebas no se alcanzaron nunca valores superiores al 60% de utilización de la CPU”.

## Volúmenes, capacidades y ratios

Como el ancho de banda utilizado, transacciones por segundo, número de usuarios, etc.



Existen dos tipos de pruebas para verificar el rendimiento del sistema:

### PRUEBAS DE CARGA

Las pruebas de carga se realizan con el objetivo de poder identificar la capacidad máxima de funcionamiento de una aplicación, así como los posibles cuellos de botella que puedan interferir en su funcionamiento a plena capacidad.

Para ello se somete a la aplicación a una carga de solicitudes que se considera puede ser la máxima real que soportará la aplicación.

### PRUEBAS DE ESTRÉS

Este tipo de pruebas tienen como objetivo comprobar la robustez, la disponibilidad y la fiabilidad de una aplicación cuando es sometida a condiciones extremas.

Lo que se pretende comprobar es cómo se comportará la aplicación cuando sea sometida a dichas condiciones extremas, que no tienen que coincidir con las esperadas en el entorno real. Es decir, si bajo estas circunstancias se provocarán denegaciones de servicio, largos tiempos de espera, se lanzarán mensajes de error, se corromperán los registros de la base de datos, etc.

# Ámbitos de aplicación de los distintos tipos de pruebas

---

## Pruebas de software

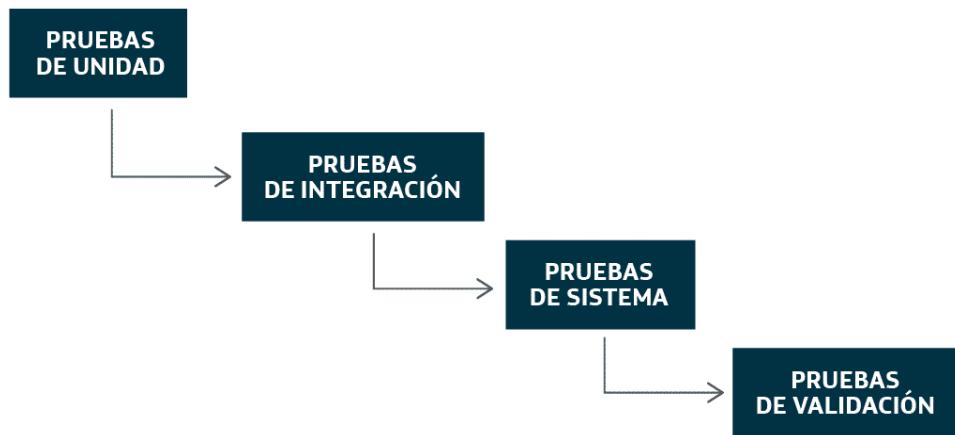
Es importante que los analistas se esfuerzen por elaborar pruebas de software desde el inicio de su desarrollo y que las vayan ampliando conforme el proyecto aumente. **Cuanto más tiempo se tarde en detectar un problema, mayor será el coste de la solución a aplicar.**

Este hecho no se debe olvidar cuando se habla de pruebas funcionales y es fundamental en el caso de las pruebas de rendimiento, ya que el ámbito de aplicación de las mismas es de principio a fin (estas se pueden ejecutar en el desarrollo inicial de la aplicación: son las conocidas pruebas de rendimiento tempranas).

El ámbito o destino de las pruebas software puede variar en tres niveles:

- **Un único módulo:** son las pruebas unitarias (funcionamiento aislado de módulos).
- **Un grupo de módulos:** son las pruebas de integración (combinando los módulos ya probados con aquel que se acaba de integrar).
- **Un sistema completo:** son las pruebas de sistema (el ámbito es el sistema en su conjunto).

## Pruebas de componentes



Las pruebas de componentes engloban las pruebas de unidad o unitarias, así como las pruebas de integración de componentes.

Para este tipo de pruebas el analista debe centrarse en los módulos que conforman el sistema.

### Pruebas de unidad

Las pruebas de unidad se realizan sobre cada módulo del software de manera independiente.

Tipos:

- **Caja negra:** conociendo la función específica para la que fue diseñado el producto. Enfoque funcional.
- **Caja blanca:** conociendo el funcionamiento del producto. Enfoque estructural.
- **Conjetura de errores:** se listan los errores comunes y se diseñan casos de prueba en función de estos.
- **Enfoque aleatorio:** se simula la entrada de datos en el sistema con generadores de pruebas.

## Pruebas de integración

En las pruebas de integración se prueban los componentes unidos para comprobar su correcto funcionamiento, puesto que al combinarlos podrían no generar la función deseada. La integración se hará de manera incremental, de menos a más.

Tipos:

- **Pruebas de regresión:** buscan reducir los efectos colaterales de añadir un nuevo componente al sistema. Hay 3 clases: pruebas sobre todas las funciones, sobre las afectadas por el cambio o sobre los componentes que han cambiado.
- **Pruebas de humo:** buscan cubrir la mayor parte de la funcionalidad del sistema. Es una estrategia de integración continua.

## Pruebas de sistema

Las pruebas de sistema son aquellas que se realizan cuando el software funciona como un todo. Para este bloque se realizan tanto pruebas de integración del sistema como pruebas de validación.

Tipos:

- **Prueba de recuperación:** se fuerza el fallo del sistema para verificar que sabe recuperarse del error en un tiempo determinado (MTTR).
- **Prueba de seguridad:** busca comprobar que el sistema sabe protegerse de accesos indeseados o ilegales.
- **Prueba de resistencia (test stress):** se aplica para ver cómo se comporta el sistema ante situaciones extremas.
- **Prueba de rendimiento:** sirve para probar el rendimiento del software en tiempo de ejecución en un sistema integrado.

## Pruebas de validación

Existen dos tipos de pruebas de validación:

- **Revisión de la configuración:** también llamada “auditoría”. Comprueba que todos los elementos de la configuración SW se han desarrollado bien.
- **Pruebas alfa y beta:** son pruebas de aceptación, para ver cómo el usuario utiliza realmente el programa.
  - **Alfa:** la realiza un cliente en un entorno controlado. El desarrollador está presente.
  - **Beta:** la realizan los usuarios finales en sus lugares de trabajo. El desarrollador no está, se le envían informes.

# Planificación

---

La planificación de las pruebas implica realizar una programación y una estimación de recursos que deberán ser tenidos en cuenta antes de realizar un proceso de pruebas del sistema.

También serán contemplados aspectos como el establecimiento de los estándares asociados a cada uno de los procesos y la descripción de las pruebas que se van a realizar.

La planificación de las pruebas es un proceso especialmente **imprescindible en desarrollos de grandes sistemas de software** y, aunque el proceso de prueba se realice sobre pequeños desarrollos, siempre será aconsejable realizar un documento donde quede reflejada la planificación del proceso de pruebas.

Aunque un plan de pruebas variará dependiendo del tipo de proyecto y de la organización que participa en la prueba, existen una serie de aspectos fundamentales que deben ser contemplados, sobre todo en planes de pruebas de grandes sistemas de software, como son:

## El proceso de prueba

Una descripción de cómo serán cada una de las principales fases del proceso de pruebas del sistema y que podrá estar dividido según el ámbito de la prueba en pruebas de subsistemas individuales, pruebas de interfaces externas del sistema, etc.

**Requisitos de trazabilidad**

Las pruebas deben planificarse de modo que se aseguren que todos los requisitos se prueban individualmente.

**Elementos a probar**

En donde se especifican todos los elementos del proceso de software que van a ser probados.

**Cronograma de pruebas**

Un calendario de pruebas y asignación de recursos en general que esté coordinado con el cronograma general de desarrollo del proyecto.

# Estructura de un plan de pruebas

---

---

Para que un plan de pruebas resulte efectivo debe componerse de las siguientes partes:

## **Procedimientos de registro de prueba**

El proceso de pruebas no se limita solamente a ejecutarlas, sino que los resultados obtenidos deberán registrarse de forma sistemática. Además, deberá ser posible la auditoría del proceso de prueba para comprobar que se ha realizado correctamente.

## **Requisitos de hardware y software**

En esta sección se deben establecer los recursos de software necesarios y la utilización del hardware estimado para que puedan ser previstos con anterioridad por los administradores de los sistemas.

## **Plan de contingencias**

Ante imprevistos que puedan afectar al proceso de prueba, como la no disponibilidad de personal o un determinado software.

## **Definición de los casos de pruebas**

Como su nombre indica, define los casos de prueba que deben aplicarse al sistema. Estas pruebas son derivadas de la especificación de requisitos del sistema.

## Resumen

---

Has finalizado esta lección. En el siguiente documento descargable está todo el contenido que has visto.

En esta lección hemos aprendido a identificar los tipos de pruebas que podemos realizar durante el desarrollo de una aplicación.

También se ha visto la importancia de tener una estrategia o plan de pruebas para asegurar el buen funcionamiento de nuestro software.



**PROEDUCA**