



Ficha de ayuda: Análisis de requisitos

El **análisis de requisitos** es la primera etapa de las pruebas. Durante esta etapa, un o una QA engineer estudia los requisitos de la aplicación para entender lo que se espera del producto que está probando.

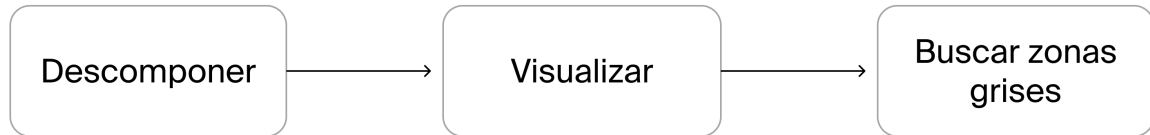
Los **requisitos de la aplicación** describen las principales características y funcionalidades de la aplicación. Tales requisitos son cruciales en las pruebas, pues nos ayudan a comprender el resultado esperado (lo que se espera que haga la aplicación y cómo lo haga).

Los objetos de prueba son partes de la aplicación que necesitan ser probadas. Estos son identificados durante el análisis de requisitos.

Para analizar los requisitos correctamente hay que seguir los siguientes pasos:

1. **Descomponer:** consiste en dividir cada requisito en bloques atómicos. Más adelante en esta lección aprenderás qué significa esto.
2. **Visualizar:** consiste en crear gráficos o diagramas sencillos (mapas mentales y diagramas de flujo) para entender cómo encajan los bloques atómicos.
3. **Buscar zonas grises:** consiste en buscar escenarios que los requisitos posiblemente no mencionen de forma explícita.

Análisis de requisitos



Cómo descomponer los requisitos

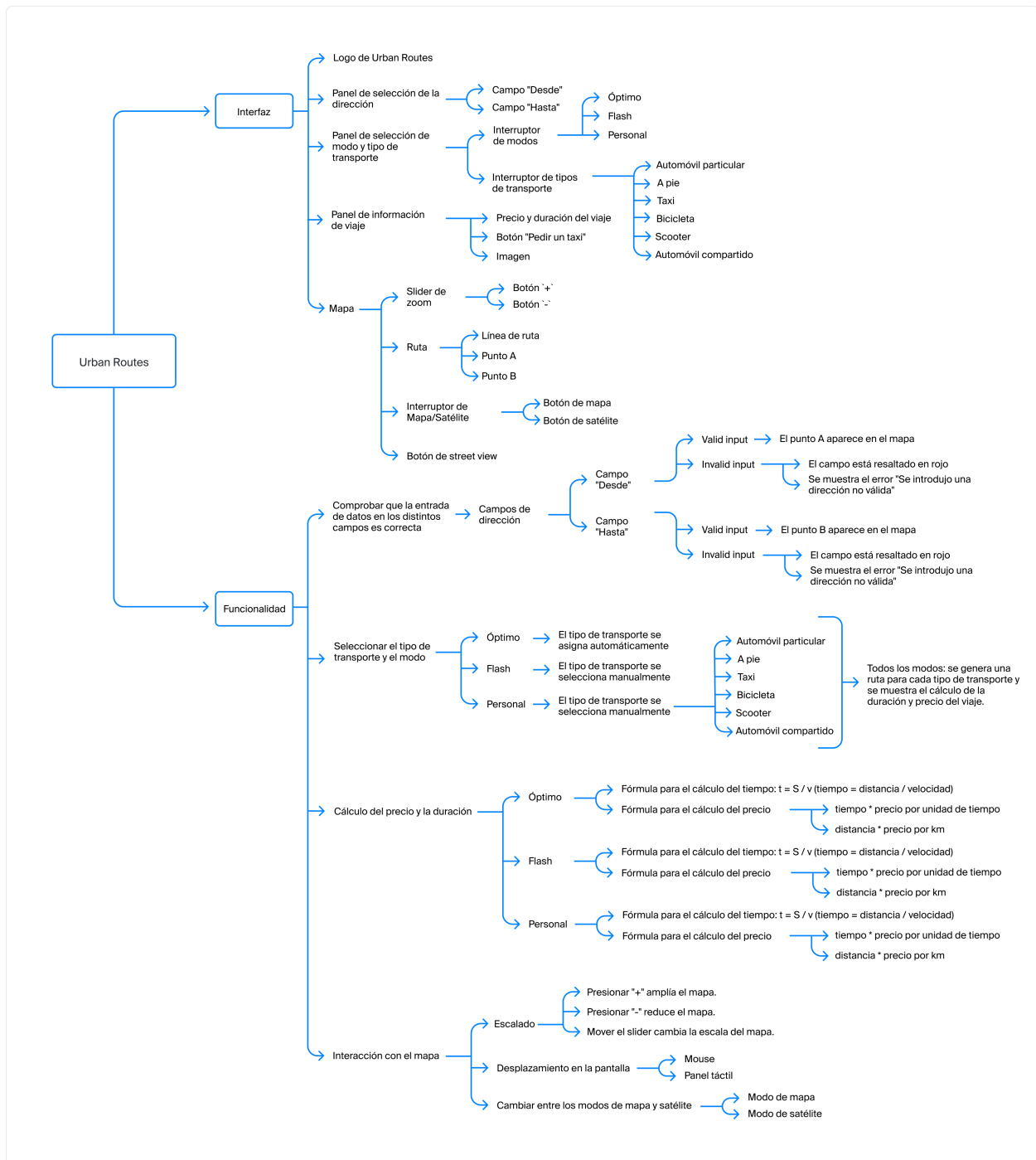
Al descomponer los requisitos puedes seguir estas tres sencillas reglas:

1. Con frecuencia, los requisitos pueden descomponerse en varias partes.
2. Los requisitos deben descomponerse en bloques atómicos.
3. Los requisitos no deben descomponerse más allá de la descripción.

Visualización: mapa mental

Un mapa mental es una técnica de visualización basada en patrones asociativos. Es una forma conveniente para que un o una tester visualice los requisitos. Con un mapa mental puedes descomponer completamente un sistema con base en diferentes criterios y, además, puedes utilizarlo como lista de comprobación.

A modo de ejemplo, a continuación te mostramos cómo puedes descomponer la página principal de Urban Routes:



Tu tarea es dividir progresivamente los requisitos en los elementos que mejor describan la forma en que percibes su funcionalidad.

El primer paso es crear el primer nivel de características.

El segundo paso es dividir cada una de las secciones en elementos hasta llegar al nivel atómico. Esto significa que los elementos no se pueden descomponer en

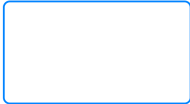
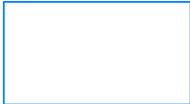

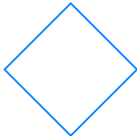

partes que afecten las características del sistema.

Visualización: diagrama de flujo

Un diagrama de flujo es una representación gráfica de un algoritmo o un proceso. Cada elemento del diagrama de flujo representa un paso en un algoritmo. Los elementos están unidos por líneas que muestran la dirección del algoritmo.

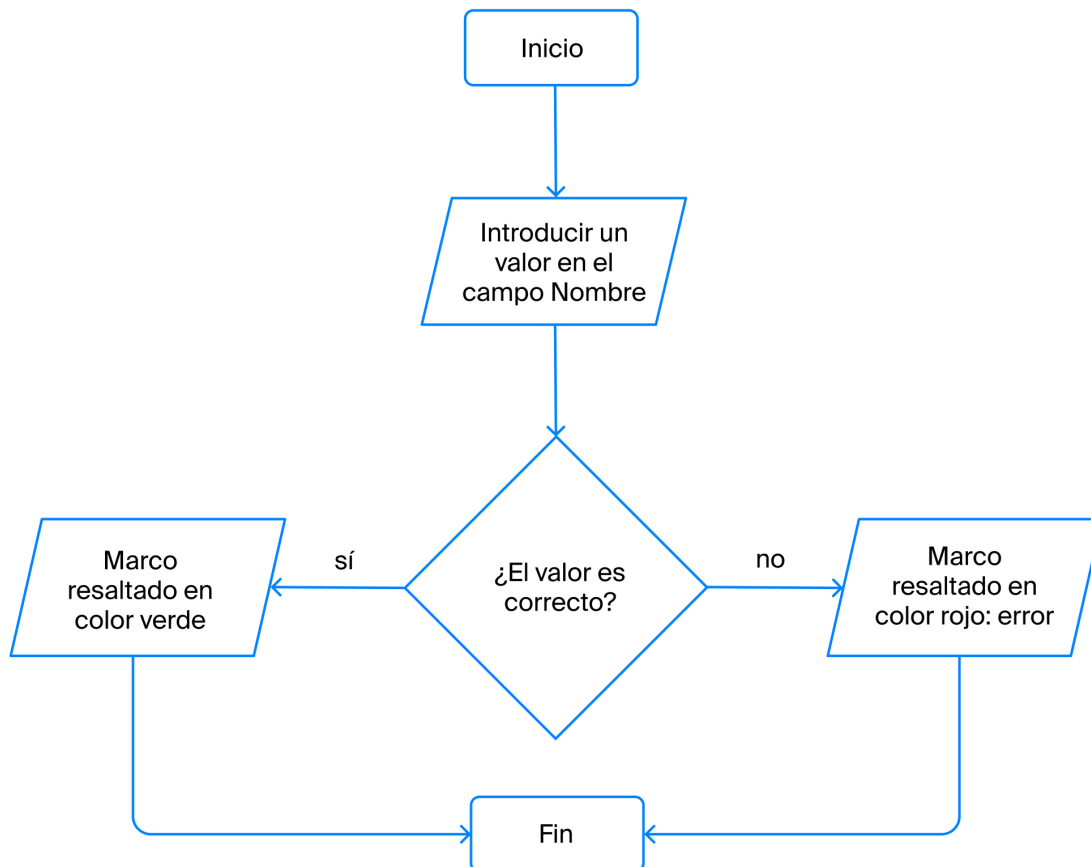
Necesitas un diagrama de flujo para visualizar:

- una funcionalidad específica;
- un algoritmo particular en un sistema.

Rectángulo redondeado		El comienzo o el final de un algoritmo
Rectángulo		Realizar una acción
Paralelogramo		Input y output
Rombo		Tomar una decisión
Flecha		Pasar a la siguiente acción

En un formulario de registro, los campos se comprueban de acuerdo con un determinado algoritmo, por ejemplo, introduciendo un nombre. Si este cumple con los requisitos, el campo se resalta con un marco de color verde, de lo contrario, el

campo se resalta con un marco de color rojo y aparece el mensaje "Entrada no válida".



Zonas grises y búsqueda de requisitos

Cuando se necesitan más detalles o información de los requisitos del producto nos encontramos con **zonas grises**.

Es posible que te encuentres con los siguientes desafíos:

- Requisitos no especificados o incompletos.

- Requisitos implícitos.
- Los diseños y los requisitos no coinciden.
- Faltan los requisitos.

Consejos para trabajar con zonas grises:

- Trata de imaginar cómo debería funcionar la aplicación.
- Ponte en el lugar del usuario: ¿es todo claro y conveniente?
- Hazte preguntas: ¿Qué propósito tiene una prueba específica? ¿Cómo funciona? ¿Con qué otros elementos está relacionada?
- Elabora un plan de pruebas y piensa qué tipo de pruebas se necesitan para esa función. Esto hará que los requisitos necesarios sean más claros.

A quién preguntarle cuando te encuentres con zonas grises:

- Área de desarrollo: pregunta sobre la implementación de los requisitos en el código y la lógica de la funcionalidad del sistema según la descripción del requisito.
- Gerencia: pregunta sobre el concepto del producto, las expectativas del usuario y del cliente y los escenarios de interacción con el producto.
- Área de diseño: pregunta sobre los escenarios de casos de uso del producto y los diseños.

Pruebas funcionales y no funcionales

Una **prueba funcional** tiene como objetivo comprobar la funcionalidad de un producto de software. En esencia, un o una QA engineer debe asegurarse de que todo funcione como se describe en los requisitos.

Una prueba funcional podría analizar estos aspectos:

- **La lógica de un sistema:** por ejemplo, probar la lógica que determina cómo se calcula el costo de un viaje.
- **La interacción del usuario con un servicio:** cómo cambia la escala de un mapa cuando haces clic sobre él, por ejemplo.

- **La visualización de datos:** por ejemplo, cómo se le muestra la información de un viaje al usuario.

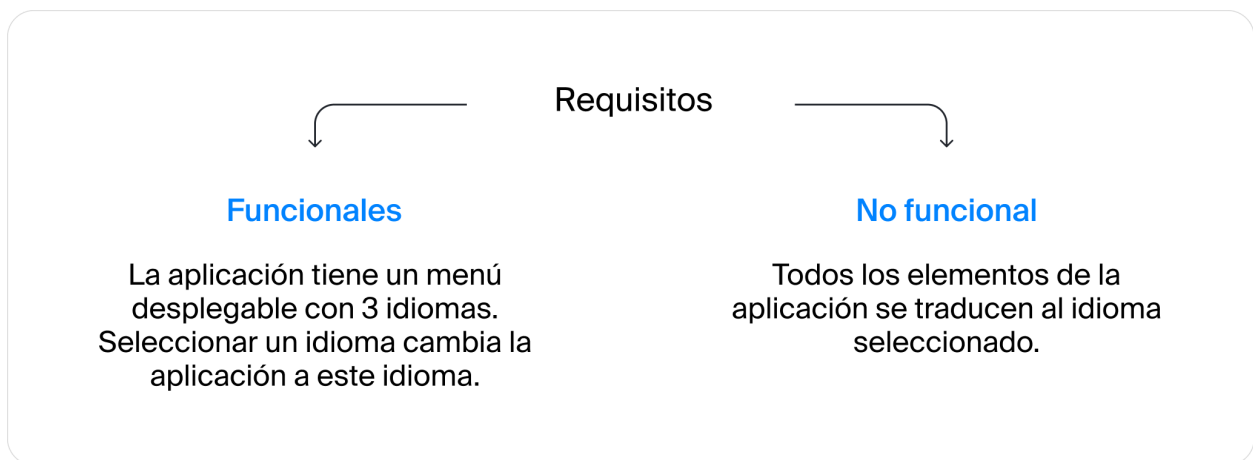
Un **requisito funcional** podría ser algo así: "al llenarse el campo 'Hasta' y hacer clic en el botón 'Enviar', la aplicación debe enviar un correo electrónico a la dirección indicada". Este requisito describe lo que la aplicación debe ser capaz de hacer o, en otras palabras, describe su funcionalidad.

Una **prueba no funcional** verifica las características de un sistema. La documentación del producto también suele contener los **requisitos no funcionales**, que se refieren a aquellas características del producto que no afectan directamente su funcionalidad.

Digamos que decidimos actualizar nuestra aplicación Urban Routes con una nueva función: un botón para cambiar de idioma para que nuestra aplicación pueda ser utilizada fácilmente en diferentes países.

Para verificar que la aplicación cumple estos requisitos, necesitamos comprobar lo siguiente:

- Si la aplicación cambia al idioma seleccionado (prueba funcional).
- Si todos los elementos son traducidos al idioma seleccionado (prueba no funcional).



Los principales tipos de pruebas no funcionales son las **pruebas de seguridad** y las **pruebas de rendimiento**.

Algunos proyectos también requieren de **pruebas de usabilidad** y **pruebas de localización**, las cuales comprueban, respectivamente, que es fácil interactuar con el software y que el texto se tradujo de forma correcta.