«Talento Tech»

Testing QA

Clase 05

Buenos Aires Ciudad

Clase 5: Tipos de Pruebas, Test Cases y su Aplicación en un Proyecto Ágil

Temario

1. **Diseño de Test Cases según el tipo de prueba**
    - ¿Qué es un Test Case?
    - Componentes clave de un Test Case.
2. **Clasificación de pruebas:**
    - Pruebas funcionales vs. no funcionales
    - Pruebas estáticas vs. dinámicas
    - Pruebas manuales vs. automatizadas
3. **Pruebas exploratorias:**
    - ¿Qué son y cuándo aplicarlas?
    - Beneficios dentro del ciclo de desarrollo ágil
4. **Planificación de pruebas en un sprint:**
    - Estrategia de ejecución de pruebas según el tipo de software
    - Herramientas recomendadas para la gestión de pruebas

Objetivo General

Comprender los diferentes **tipos de pruebas** dentro del ciclo de desarrollo ágil y aprender a **diseñar Test Cases** efectivos que permitan validar la calidad del software. Los estudiantes aplicarán estos conceptos en la **App del Hotel**, asegurando la cobertura adecuada de pruebas.

Test Case

Un **Test Case** es un conjunto de instrucciones detalladas que indican qué debe probarse, cómo debe probarse y cuál es el resultado esperado.

✔ ¿Por qué son importantes?

- Permiten replicar pruebas de forma **consistente y estructurada**.
- Facilitan la **detección y documentación de defectos**.
- Aseguran una **cobertura de pruebas efectiva**.

📌 Componentes Clave de un Test Case:

Campo	Descripción	Ejemplo (App del Hotel)
ID	Identificador único del Test Case.	TC-001
Título	Nombre descriptivo del Test Case.	"Validar reserva con fechas correctas"
Descripción	Explicación breve del objetivo del Test Case.	"Verificar que el usuario pueda completar la reserva seleccionando fechas válidas"
Precondiciones	Requisitos previos para ejecutar la prueba.	"El usuario debe estar logueado en la app"
Pasos	Secuencia de acciones a realizar.	1. Seleccionar check-in y check-out. 2. Escoger habitación disponible. 3. Completar datos personales. 4. Confirmar reserva.
Datos de Prueba	Valores que se ingresarán durante la prueba.	Check-in: 12/04/2024 Check-out: 15/04/2024
Resultado	Lo que debería ocurrir si el sistema funciona correctamente.	"La reserva debe completarse y el usuario debe recibir una confirmación"
Estado	Indica si la prueba pasó o falló.	"Aprobado" / "Fallido"

Explicación Detallada Campo por Campo

1) **ID:** Es un **identificador único** para cada Test Case, lo que permite rastrearlo fácilmente.

   ¿Por qué es importante?

   - Facilita la organización en planillas de test cases.
   - Permite referenciarlo en reportes de defectos.

   Ejemplo en la App del Hotel:

   - **TC-FUNC-001:** Test Case funcional para verificar reservas.
   - **TC-NF-002:** Test Case no funcional para medir tiempos de carga.

2) **Título:** Es un nombre **descriptivo** que indica qué funcionalidad se está probando.

   ¿Por qué es importante?

   - Permite identificar rápidamente de qué trata el Test Case.
   - Debe ser **claro y preciso**.

   Ejemplo en la App del Hotel:

   - "Reserva de habitación con datos válidos"
   - "Intento de pago con tarjeta inválida"

3) **Descripción:** Es una **breve explicación** del propósito del Test Case.

   ¿Por qué es importante?

   - Ayuda a otros testers a entender el objetivo de la prueba.
   - Sirve como referencia en revisiones de calidad.

   Ejemplo en la App del Hotel:

   - "Verificar que el usuario pueda completar una reserva exitosamente."
   - "Evaluar si el sistema muestra un error al ingresar una tarjeta inválida."

4) **Precondiciones:** Son los **requisitos previos** que deben cumplirse antes de ejecutar la prueba.

   ¿Por qué es importante?

   - Asegura que el entorno de prueba esté correctamente configurado.
   - Evita errores causados por condiciones incorrectas.

Ejemplo en la App del Hotel:

- "El usuario debe estar logueado en la app."
- "La base de datos debe contener habitaciones disponibles."

5) **Pasos**: Es una **lista detallada** de las acciones a seguir para ejecutar la prueba.

   ¿Por qué es importante?

   - Asegura **consistencia** en la ejecución de pruebas.
   - Permite que cualquier tester pueda replicar el proceso.

   Ejemplo en la App del Hotel:

   1. Iniciar sesión en la app.
   2. Seleccionar fechas de check-in y check-out.
   3. Elegir una habitación disponible.
   4. Completar los datos personales.
   5. Confirmar la reserva.

6) **Datos de Prueba:** Son valores específicos que se ingresarán en la prueba.

   ¿Por qué es importante?

   - Permite replicar la prueba con los mismos datos en diferentes entornos.
   - Asegura **coherencia en la validación** de resultados.

   Ejemplo en la App del Hotel:

   - **Check-in:** 10/04/2024
   - **Check-out:** 15/04/2024
   - **Tarjeta de crédito válida:** 4111 1111 1111 1111

7) **Resultado Esperado:** Es lo que **debería ocurrir** si el sistema funciona correctamente.

   ¿Por qué es importante?

   - Sirve como criterio para determinar si la prueba **fue exitosa o falló**.
   - Facilita la detección de **errores y desviaciones**.

   Ejemplo en la App del Hotel:

   - "La reserva se confirma y el usuario recibe un correo con los detalles."
   - "El sistema muestra un mensaje de error si la tarjeta es inválida."

8) **Resultado Obtenido:** Es lo que realmente ocurrió al ejecutar el Test Case.

   ¿Por qué es importante?

   - Permite comparar con el **resultado esperado** y detectar defectos.
   - Se usa para **documentar fallos** en caso de que algo no funcione correctamente.

   Ejemplo en la App del Hotel:

   - "La reserva se confirmó correctamente. (Aprobado)"
   - "El sistema no mostró el error esperado. (Fallido)"

9) **Estado:** Es el resultado final de la prueba.

Opciones Comunes:

✅ **Aprobado:** La prueba pasó sin errores.
❌ **Fallido:** Se encontró un error o comportamiento inesperado.
🟡 **Pendiente:** La prueba aún no se ejecutó.

Ejemplo en la App del Hotel:

- "La reserva se confirmó correctamente." → Estado: Aprobado ✅
- "El sistema permitió pagar con una tarjeta inválida." → Estado: Fallido ❌

Conclusión

¿Por qué es importante documentar correctamente los Test Cases?

✔ Facilita la organización y ejecución de pruebas.
✔ Permite replicar pruebas con consistencia.
✔ Ayuda a identificar errores y mejorar la calidad del software.

Pruebas en QA

Son un conjunto de procesos y metodologías utilizadas para garantizar la calidad del software. Su objetivo es identificar defectos, errores o problemas en un sistema antes de que llegue a producción, asegurando que funcione correctamente y cumpla con los requisitos definidos.

Clasificación de Pruebas

Para garantizar la calidad de una aplicación, es fundamental conocer los distintos **tipos de pruebas** y aplicarlas de manera efectiva.

Pruebas funcionales vs. no funcionales

✔ **Pruebas Funcionales:** Verifican que el software cumpla con los requerimientos definidos, centrándose en las funcionalidades esperadas.

✔ **Pruebas No Funcionales:** Evalúan aspectos como el rendimiento, la usabilidad, la seguridad o la compatibilidad.

Ejemplo en la APP del Hotel:

- **Funcional:** Verificar que los usuarios puedan registrarse y reservar una habitación con éxito.
- **No Funcional:** Medir el tiempo de carga del sistema al procesar una reserva con múltiples habitaciones.

Pruebas estáticas vs. dinámicas

✔ **Pruebas Estáticas:** Se realizan sin ejecutar el código, como la revisión de documentación o análisis de requerimientos.

✔ **Pruebas Dinámicas:** Se ejecuta el software para verificar su comportamiento en tiempo real.

Ejemplo en la APP del Hotel:

- **Estática:** Revisar la documentación de los requisitos del sistema antes de programar la funcionalidad de check-in online.
- **Dinámica:** Ejecutar la funcionalidad de check-in online en distintos dispositivos para validar su correcto funcionamiento.

Pruebas manuales vs. automatizadas

✔ **Pruebas Manuales:** Se ejecutan sin el uso de herramientas de automatización, evaluando la experiencia del usuario.

✔ **Pruebas Automatizadas:** Se implementan scripts para validar funcionalidades de manera repetitiva y eficiente.

Ejemplo en la APP del Hotel:

- **Manual:** Un tester reserva una habitación y verifica si recibe el correo de confirmación.
- **Automatizada:** Un script de Selenium ejecuta 100 reservas simultáneamente para validar que el sistema no colapse.

Pruebas Exploratorias

Las pruebas exploratorias permiten descubrir errores que no estaban contemplados en los test cases.

¿Cuándo aplicarlas?

- Cuando no se dispone de documentación detallada.
- Para evaluar funcionalidades nuevas en entornos ágiles.

Beneficios:

- Permiten identificar defectos inesperados.
- Mejoran la comprensión del software.

Ejemplo en la APP del Hotel:
Un tester explora la funcionalidad de check-out, probando combinaciones no previstas, como intentar cerrar sesión mientras se finaliza una reserva.

Planificación de Pruebas en un Sprint

Para optimizar el testing en entornos ágiles, es necesario **planificar la ejecución de pruebas** de forma estructurada.

✔ **Estrategia de ejecución:**

1. **Identificar funcionalidades críticas** a testear en el sprint.
2. **Determinar qué tipos de pruebas aplicar** (funcionales, exploratorias, automatizadas, etc.).
3. **Definir prioridades** según impacto y riesgo.
4. **Utilizar herramientas** para la gestión de pruebas.

Ejemplo en la APP del Hotel:
Si en un sprint se desarrolla una nueva funcionalidad de pago en línea, el equipo debe priorizar pruebas funcionales, de usabilidad y de seguridad antes del lanzamiento.

Realicemos una clasificación y planificación de pruebas:

App del Hotel:

Paso 1: Clasificar Pruebas para el Módulo de Carga de CV.

En este paso, en lugar de analizar la carga de CV en **Talento Lab**, aplicamos el concepto al **formulario de reserva de la App del Hotel**.

Objetivo: Determinar qué pruebas funcionales y no funcionales aplicar para validar correctamente el formulario de reserva.

📌 **User Story Base:**
"Como usuario, quiero llenar un formulario de reserva para completar mi solicitud de alojamiento."

Pruebas Funcionales (Evalúan si el sistema hace lo que debería hacer)

- Verificar que el usuario pueda ingresar las fechas de check-in y check-out.
- Asegurar que el sistema calcule correctamente el total a pagar según la cantidad de noches y el tipo de habitación.
- Validar que solo se aceptan fechas válidas (ejemplo: no permitir una fecha de salida anterior a la fecha de ingreso).
- Confirmar que el sistema muestra un mensaje de error si faltan datos obligatorios.
- Revisar que el botón "Reservar" solo se active cuando todos los campos sean completados correctamente.

Pruebas No Funcionales (Evalúan cómo se comporta el sistema más allá de sus funcionalidades básicas)

- **Pruebas de rendimiento:** Verificar que el formulario de reserva cargue en menos de **3 segundos**.
- **Pruebas de compatibilidad:** Asegurar que el formulario funcione en **PC, tablets y móviles** con distintos navegadores.
- **Pruebas de usabilidad:** Evaluar si el diseño del formulario es intuitivo y fácil de completar.
- **Pruebas de seguridad:** Asegurar que los datos ingresados por el usuario estén encriptados.

Paso 2: Ejecutar Pruebas Exploratorias sobre el Formulario de Reserva.

Objetivo: Descubrir posibles errores no contemplados en los test cases iniciales probando el sistema sin un guión predefinido.

✔ **Escenario 1:** Intentar reservar sin completar algún campo obligatorio.

- **Resultado esperado:** El sistema debe mostrar un mensaje de error indicando qué campo falta.
- **Resultado obtenido:** Si el sistema permite avanzar sin completar los datos, hay un defecto que debe ser reportado.

✔ **Escenario 2:** Probar con fechas fuera de rango (ejemplo: año 2050).

- **Resultado esperado:** El sistema debe validar que las fechas ingresadas sean realistas.
- **Resultado obtenido:** Si permite una reserva en el año 2050 sin restricciones, se debe reportar como defecto.

✔ **Escenario 3:** Introducir caracteres inválidos en el campo de número de personas.

- **Resultado esperado:** El sistema debe permitir solo números.
- **Resultado obtenido:** Si permite ingresar letras, hay un error en la validación.

Paso 3: Crear un Plan de Pruebas en Hoja de Cálculo.

Objetivo: Documentar y priorizar los test cases más importantes en un sprint.

 **Ejemplo de Plan de Pruebas**

Conclusión

La planilla de pruebas nos permite estructurar, organizar y priorizar los test cases de forma clara y eficiente.

- ✔ Permite rastrear cada prueba con un ID único.
- ✔ Define el escenario que se está evaluando.
- ✔ Clasifica la prueba como funcional, no funcional o exploratoria.
- ✔ Establece la prioridad según el impacto en la aplicación.
- ✔ Describe el resultado esperado para validar el comportamiento del sistema.
- ✔ Registra el estado de la prueba para saber si pasó o falló.

Reflexión Final

“No se trata solo de probar, sino de probar con estrategia.”

Los testers deben seleccionar el tipo de prueba adecuado para cada escenario, priorizando la calidad sin retrasar el desarrollo. En equipos ágiles, la **colaboración y la flexibilidad** son clave para asegurar que el producto final cumpla con las expectativas del usuario.

Material Complementario

- 📖 **ISTQB - Foundation Level Syllabus:** Capítulo sobre pruebas funcionales y no funcionales.
- 🎥 Video: Exploratory Testing in Agile

Ejercicio Práctico

¡Seguimos trabajando en el Proyecto!

El equipo de **Talento Lab** enfrenta un sprint con múltiples funcionalidades nuevas. **Silvia** les explica que no basta con ejecutar pruebas funcionales, sino que deben combinar distintos enfoques.

Matías propone realizar **pruebas exploratorias** en los módulos recién desarrollados y definir qué escenarios requieren **pruebas automatizadas** en el futuro.

Sitio Web TechLab: https://talentolab-test.netlify.app/

1) **Diseñar Test Cases:**
   - Crear al menos 3 Test Cases para validar diferentes escenarios.
2) **Clasificar pruebas:**
   - Analizar **una feature** de las seleccionadas anteriormente.
   - Determinar qué pruebas funcionales y no funcionales aplicar.
   - Justificar la elección en un documento.
3) **Ejecutar pruebas exploratorias:**
   - Diseñar y documentar un conjunto de **pruebas exploratorias** sobre la **feature** .
   - Identificar posibles defectos o mejoras.
4) **Plan de pruebas en un sprint:**
   - Crear un **plan de pruebas en hoja de cálculo**, priorizando las pruebas según criticidad.

Ejemplo de Plan de Pruebas


Buenos Aires
aprende
Agencia de Habilidades para el Futuro
Buenos
Aires
Ciudad