Anotaciones sobre Testing

**Tipos de metodología:**

**Metodologías secuenciales:**

Son metodologías o ciclo de vida de desarrollo de software las cuales son secuenciales, es decir, cada etapa requiere de que todas las anteriores sean cumplidas y se ejecutan una después de otra, tienen como característica principal que en ellas solo hay UNA SOLA ENTREGA al cliente, la cual se le hace de una vez por todas cuando el software ya esta totalmente terminado.

**Ventajas:**

* Es fácil de entender

**Desventajas:**

* Al hacerse una SOLA ENTREGA cuando el Software ya está completamente terminado, si al cliente no le gusta o se debe hacer algún cambio, se deberá repetir el proceso desde 0, lo cual sería muy costoso y trabajoso.
* Al hacerse la entrega solamente cuando el Software ya esté completamente terminado, el cliente tarda más tiempo en ver resultados.
* No se obtiene Feedback Continuo del cliente, o incluso no se obtiene Feedback alguno hasta que ya el software está terminado.

**Uso optimo:**

* Cuando tenemos garantías y 100% asegurado que no habrá NINGÚN TIPO DE CAMBIO en los requerimientos

**Waterfall o Cascada:** Es una metodología secuencial en la cual los pasos son secuenciales y no hay ningún tipo de solapamiento entre las etapas, los pasos de Testing no se ejecutan en paralelo con los pasos de Development. Consta de las siguientes etapas:

Requerimientos -> Diseño -> Desarrollo -> Testing -> Despliegue -> Mantenimiento

**V-Model:** Es una metodología secuencial en la cual, aunque las etapas también son secuenciales, los pasos de Testing se ejecutan en paralelo con otras etapas:

Requerimientos <------------------------------------------------ > Pruebas de aceptación

Especificaciones <------------------------------------------ > Pruebas de sistema

Diseño arquitectónico <------------------------------ > Pruebas de integración

Diseño detallado <------------------- > Pruebas unitarias

Desarrollo o codificación

Para cada etapa se desarrolla escenarios de pruebas, las cuales van por orden de complejidad, de la más sencilla a la más compleja:

**Pruebas unitarias:** Son pruebas de unidades pequeñas de código como funciones o clases, en el contexto de las pruebas se puede decir que es cuando probamos una función muy específica como por ejemplo verificar la validez de una contraseña en un registro de usuarios.

**Pruebas de integración:** Son pruebas en las cuales verificamos que distintos componentes del software (Base de Datos, UI, o incluso otros módulos o componentes) trabajan en conjunto de manera correcta. Hay dos tipos de pruebas de integración:

* **De Componentes:** Es generalmente hecha por un desarrollador, en la cual prueba que los diferentes componentes o porciones de código que programo funcionen correctamente en conjunto.
* **De Sistemas:** Es generalmente hecha por el tester, en la cual prueba el frontend, el backend y la base de datos, los cuales, en este contexto, cada uno de ellos son considerados un sistema.

**Pruebas funcionales:** Son pruebas en las cuales verificamos que las funcionalidades funcionen de manera correcta desde la perspectiva de un usuario final.

**Pruebas de sistema:** Son pruebas en las cuales verificamos el funcionamiento del sistema como un todo, tratando de encontrar la mayor cantidad de defectos posibles. Este es el tipo de pruebas más importante de las mencionadas ya que aquí es donde está la mayor parte del trabajo de un QA Tester. Aquí es donde verificaremos el funcionamiento del sistema “End To End” es decir, al completo, desde una perspectiva similar a la de un usuario final, ya que el sistema ya debería poder generar información que será utilizada por el usuario.

**Pruebas de aceptación:** Son pruebas en las que nos aseguramos que el sistema está listo para el despliegue, verificando que el rendimiento del software. Mientras que las pruebas de sistema están enfocadas en encontrar bugs o errores, las pruebas de aceptación están enfocadas en el rendimiento del sistema como un todo. Hay ciertos tipos de pruebas de aceptación que involucran a los clientes para que utilicen el software mientras aún está en fase de pruebas (versiones Alpha y Beta).

**Metodologías ágiles:**

**Scrum:** Es una metodología ágil organizada por sprints, en la cual tenemos un Scrum Team formado por el Product Owner (que sustituye al Business Analyst en las metodologías secuenciales), el Scrum Master y los Developers (todo el equipo que no forma parte de los dos roles anteriores).

**Stakeholders:** Son los interesados en el proyecto, como los usuarios finales o los clientes.

**Product Owner:** Es el representante de los stakeholders dentro del equipo, escribe los ítems o user stories del Product Backlog y es el único con autoridad para cancelar o cambiar un sprint.

**Scrum master:** Es el “coach” o “manager” de un equipo Scrum, que gestiona el equipo.

**Developers:** Es el resto del equipo los cuales participan en todo el proceso de llevar el proyecto a la realidad.

A grandes rasgos, Scrum consiste en tomar un Product Backlog (lista de requerimientos), seleccionar algunos ítems de esta, y en base a estos ítems seleccionados trabajar el “mini-proyecto” en un Sprint, que es una ventana de tiempo que va desde 2 a 4 semanas, en la que al final de esta se entregará un incremento o entregable al cliente con funcionalidad parcial que alcanza el Minimo Producto Viable, gracias a lo cual podrá dar feedback y en caso de que se requiera algún cambio no tener que empezar todo el proceso desde cero.

A grandes rasgos, Scrum funciona así:

Requerimientos (Product Backlog)

|

Seleccionar los ítems o user stories más prioritarios (Sprint backlog)ç

|

Planeación del Sprint

|

Sprint (De 2 a 4 Semanas)

Daily Scrums (Reuniones de máximo 15 minutos diarias para saber que ha hecho ayer, que hará hoy y los problemas que ha tenido cada developer)

|

Review del Sprint (8 horas máximo) para analizar el resultado del Sprint y presentar el entregable a los stakeholders

|

Sprint Retrospective para buscar maneras de incrementar la efectividad y calidad en los Sprints futuros

Una vez terminado un Sprint el proceso se repite inmediatamente.

**Modelos de desarrollo de Software:**

**Secuencial:** Es el modelo que ya vimos anteriormente, cada etapa va después de la anterior y se entrega al cliente una sola vez: Al final del proyecto.

**Incremental:** Se entrega al cliente pequeñas funcionalidades especificas continuamente, sin embargo, por sí solas no son útiles o usables. Por ejemplo, se entrega la funcionalidad de cargar fotos, otra de registrar usuario, etc.

**Iterativa:** Se entrega al cliente un “mini-software” completo, que requirió un ciclo de vida de desarrollo completo, se entrega una aplicación “completa” solo con las funcionalidades más críticas, es el modelo usado por las metodologías ágiles como Scrum. Una vez entregada, pues basado en el feedback recibido, se incrementa la aplicación añadiéndole mejoras y las funcionalidades faltantes. Un ejemplo práctico de esto, por así decirlo, son las diferentes versiones o fases de los juegos: Fase Alpha, Fase Beta, etc.

**Proceso de Pruebas (Test Process):**

**Planeación:** Escribimos nuestro plan de pruebas

**Monitoreo y control:** Verificamos nuestro progreso esperado con el progreso real que hemos tenido según el plan.

**Análisis:** Analizamos los requerimientos para escribir nuestros escenarios de prueba y nuestros casos de prueba.

* **Casos de pruebas:** Es una descripción detallada, paso por paso y diciéndonos COMO debemos probar lo que debemos, es más detallado que los escenarios de prueba y requieren más tiempo para ejecutarse. Se usa generalmente cuando tenemos más tiempo para ejecutarlos.

Ejemplo explicado aquí: https://www.youtube.com/watch?v=Yg9P2KnUiEM

* **Escenarios de pruebas:** Es cualquier funcionalidad del software que puede ser probada, son una descripción simple y general de lo QUE vamos a probar, generalmente se escriben en una oración simple. Un escenario de prueba es la descripción de una serie de casos de prueba relacionados a una funcionalidad. Se usa generalmente cuando tenemos poco tiempo, como cuando trabajas con metodologías agiles, cuando las funcionalidades cambian mucho y cuando ya estas ambientado en el proyecto. OJO: A DIFERENCIA DE LOS CASOS DE PRUEBA, EN LOS ESCENARIOS DE PRUEBAS NO ESPECIFICAMOS EL RESULTADO ESPERADO.

Como dijimos antes, suelen requerir o conformarse por varios casos de prueba:

Ejemplo: Hacer una búsqueda de un producto en MercadoLibre

Para esto, primero debemos verificar una serie de casos o sub-funcionalidades que el cliente esperaría al hacer una búsqueda en Mercado Libre:

* 1. Poder entrar a la página principal
  2. Ver una barra de búsqueda
  3. Obtener sugerencias de búsqueda al hacer clic en la barra de búsqueda
  4. Al hacer clic en el botón de buscar obtener un listado de productos que sean lo que se buscó en la barra de búsqueda.

En resumen, podemos decir que una funcionalidad (por ejemplo, registro de usuario) puede tener múltiples escenarios de prueba (por ejemplo la validación del campo de correo electrónico, o del campo de contraseña), y que un escenario de prueba, puede contener multiples casos de prueba (por ejemplo, un caso de prueba (válido) en el que el correo electrónico es válido, otros (inválidos) en el que el formato del correo electrónico no tiene el .com, o no tiene el @, otro caso de prueba en el que se introduzca un correo electrónico ya registrado, etc.)

Los escenarios de prueba son un conjunto de condiciones o circunstancias que representan situaciones, que podrían ser válidas o inválidas (casos de prueba), bajo las cuales se prueban una parte especifica de una funcionalidad.

**Diseño:** Diseñamos nuestros casos de pruebas.

**Implementación:** Organizamos nuestros casos de prueba en “Test Suites”, diseñamos nuestro ambiente de pruebas en el cual ejecutaremos las pruebas (por ejemplo, un emulador, un navegador como Chrome o Bing, un Sistema Operativo como Windows o Android, etc), SE DEBE INTENTAR HACER LAS PRUEBAS EN UNA AMBIENTE IGUAL O AL MENOS SIMILAR AL AMBIENTE DE PRODUCCIÓN, ES DECIR, SI EL CLIENTE UTILIZARÁ EL SOFTWARE EN WINDOWS, SE DEBERÍAN HACER LAS PRUEBAS EN WINDOWS, SI USARA EL SOFTWARE EN UN CELULAR DE 7 PULGADAS CON ANDROID 13.5 SE DEBERÍAN HACER LAS PRUEBAS EN UN CELULAR O EMULADOR DE ANDROID 13.5 CON 7 PULGADAS DE PANTALLA.

* **Test Suites:** Son colecciones organizadas de casos de prueba que se agrupan y se ejecutan de forma conjunta para probar una funcionalidad, componente o módulo de un software. Se diferencian de los escenarios de prueba en que los escenarios son simplemente descripciones de alto nivel de una situación específica que se debe probar.

**Ejecución:** Ejecutamos los distintos tipos de pruebas que necesitemos ejecutar (pruebas de rendimiento, regresión, etc.), si encontramos defectos los reportamos

**Terminación:** Terminamos el trabajo escribiendo reportes de progreso de pruebas y reportes de resúmenes de pruebas.

**Tipos de pruebas:**

Los dos tipos de pruebas principales son:

**Según funcionalidad:**

**Funcionales:** Son las pruebas MÁS COMUNES, en las que verificamos y nos aseguramos de que el software cumple con las funcionalidades esperadas, por ejemplo, poder registrarme e iniciar sesión en un software, y entrar a un perfil que contenga fotos que solo yo y mis contactos podamos ver.

Se pueden resumir en una respuesta de Sí o No: ¿Cumple el software con X requerimiento? ¿Sí o No?

**No Funcionales:** Son las pruebas en las que verificamos otros aspectos al margen de la funcionalidad, generalmente no son preguntas que se pueden responder con un sí o un no, sino que las pruebas arrojan ciertos resultados, que en base al tipo de software que estemos construyendo y a los requerimientos entonces interpretaremos y podemos decir si los resultados son buenos o malos, las pruebas no funcionales cubren aspectos como:

* **Seguridad:** Son pruebas de seguridad del software, que tantas vulnerabilidades tiene, etc.
* **Rendimiento:** Son pruebas del rendimiento del software, como los recursos que consume para hacer alguna operación o por ejemplo, el tiempo de respuesta de una aplicación, el estándar aceptado puede variar de acuerdo al tipo de aplicación que estemos desarrollando, ya que por ejemplo, 1 segundo de respuesta puede ser bueno para una funcionalidad o aplicación especifica como una red social, pero ese mismo tiempo podría ser un tiempo de respuesta pésimo para por ejemplo un chat en tiempo real o un videojuego online.
* **Usabilidad:** Son pruebas de que tan fácil de utilizar es el software, que tan atractiva e intuitiva es el UI, etc.
* **Compatibilidad:** Son pruebas a la capacidad del software de funcionar adecuadamente en diferentes entornos como en los diferentes navegadores existentes (Chrome, Bing, Firefox, etc.), sistemas operativos, dispositivos y configuraciones.

**Según el conocimiento del código:**

**Black Box Testing:** Son pruebas en las que no tenemos ningún conocimiento de cómo funciona el código del módulo o funcionalidad que estamos probando, simplemente ejecutamos la prueba, y sabemos las entradas y salidas que da el software.

**White Box Testing:** Son pruebas en las que se nos ha proporcionado el código del software o de la parte del software que estamos probando, realizamos las pruebas mirando la estructura interna del código, que pasa en el código, la API y en la base de datos mientras realizamos las pruebas. En este tipo de pruebas requerimos saber o entender el lenguaje de programación en el cual está escrito el software que estamos probando.

**Según la ejecución del programa:**

**Dynamic Testing:** Son las pruebas en las cuales ejecutamos el programa al hacer las pruebas. Es decir, este tipo de pruebas incluyen la ejecución del software.

**Static Testing:** Son las pruebas las cuales no incluyen la ejecución del software, es decir, es todo lo que hacemos fuera de la ejecución de las pruebas, como por ejemplo revisar las historias de usuario, los diseños del UI, revisar código, etc.

**Según el momento:**

**Testing:** Son las pruebas que ejecutamos inicialmente.

**Re-Testing o Confirmation Testing:** Son las pruebas que ejecutamos una vez ya ejecutamos una prueba de algún aspecto del software previamente y esta no cumplió satisfactoriamente con lo esperado o se encontraron defectos, por lo cual se devolvió a los desarrolladores para que lo reparasen. Al ejecutarse nuevamente pruebas sobre la misma funcionalidad o componente que NO paso la prueba, se está haciendo un Re-Testing para asegurarse que esta vez si se cumple con lo esperado. IMPORTANTE: EN EL RE-TESTING DEBEMOS EJECUTAR LOS MISMOS PASOS QUE HICIERON QUE SE ENCONTRARÁ EL DEFECTO O EL BUG EN EL SOFTWARE PARA ASEGURARNOS DE QUE EL PROBLEMA REALMENTE FUE RESUELTO.

**Regression Testing:** Son las pruebas que hacemos sobre toda la aplicación después de hacer algún cambio en el software como agregar alguna nueva funcionalidad, remover algo o actualizarlo. Este tipo de prueba se ejecuta para asegurarse que al agregar o hacer algún cambio en el software no se rompieron cosas que ya funcionaban.

**Smoke Testing:** Son pruebas en las que probamos rápida y superficialmente el software, probando las funcionalidades críticas o principales del software (como el inicio o arranque de la aplicación, la navegación por las diferentes vistas o páginas de la aplicación y las funcionalidades fundamentales) y buscando errores críticos y graves que impidan que el software sea útil o funcional. En caso de que el software pase la prueba de humo, se procede con otras pruebas más detalladas y minuciosas, en caso de que el software no pase la prueba de humo, se reportan los errores encontrados para que sean reparados.

**Otros tipos de pruebas:**

**Pruebas de localización (Localization Testing):** Consiste en hacer pruebas de la aplicación en otro idioma o en otro sistema de escritura (por ejemplo, árabe, japones o chino)

**Escenarios de prueba**

Son cualquier funcionalidad de la aplicación la cual puede ser testeada, son también llamados Condiciones de Pruebas o Posibilidades de Pruebas, en los escenarios de prueba capturamos escenarios u ocasiones y usos de la aplicación que podrían darle los usuarios finales una vez lanzada la aplicación.

**Como escribir escenarios de pruebas:**

1. Estudiamos y analizamos la documentación existente sobre los requerimientos del proyecto, sea esta documentación Business Requeriments Specification (BRS), Software Requirements Specificacion (SRS), Functional Requirements Specification (FRS), o incluso versiones antiguas de la aplicación o algún prototipo o cualquier cosa que pueda servir como base.
2. Aislamos y tomamos cada requerimiento, es decir, intentamos responder la pregunta “¿Qué o cuales requerimientos tenemos?”, identificamos posibles acciones que podrían ejecutar los usuarios en la aplicación (incluyendo usuarios maliciosos o malintencionados)
3. Enumeramos los escenarios de prueba que abarquen cada posible “feature” del software.
4. Discriminamos entre los escenarios de prueba para identificar los escenarios de prueba críticos o más importantes de los menos importantes.
5. Creamos nuestra Matriz de Trazabilidad (dígase, la típica tablita en la cual ponemos las informaciones como el ID, los pasos, si paso o no la prueba, etc) y ponemos en ella todos los escenarios de prueba listados, para asegurarnos que cada requerimiento este asociado a un escenario de prueba.
6. Muestra los escenarios de prueba a algún supervisor y hazlo ser revisados por los stakeholders o clientes para obtener feedback de ellos.

IMPORTANTE: LOS ESCENARIOS DE PRUEBA Y CASOS DE PRUEBA GENERALMENTE SE ESCRIBEN EN TERCERA PERSONA.

**Escribiendo escenarios de pruebas:**

**Paso 1:**

Obtenemos y analizamos los requerimientos, en este caso, analizaremos un Software Requirement Specification (SRS):

**Ejemplo de Software Requirement Specification (SRS):**

**ID:** FR3

**TITULO:** Registro de Usuario – Aplicación Móvil

**DESC:** Dado que el usuario ha descargado la aplicación móvil, el usuario debe de poder registrarse utilizando la aplicación móvil. El usuario tiene que proporcionar un nombre de usuario valido, su email y una contraseña valida. El usuario puede elegir proveer un número de teléfono utilizado frecuentemente.

El nombre de usuario tiene que ser único por cada usuario, debe de tener al menos 6 caracteres.

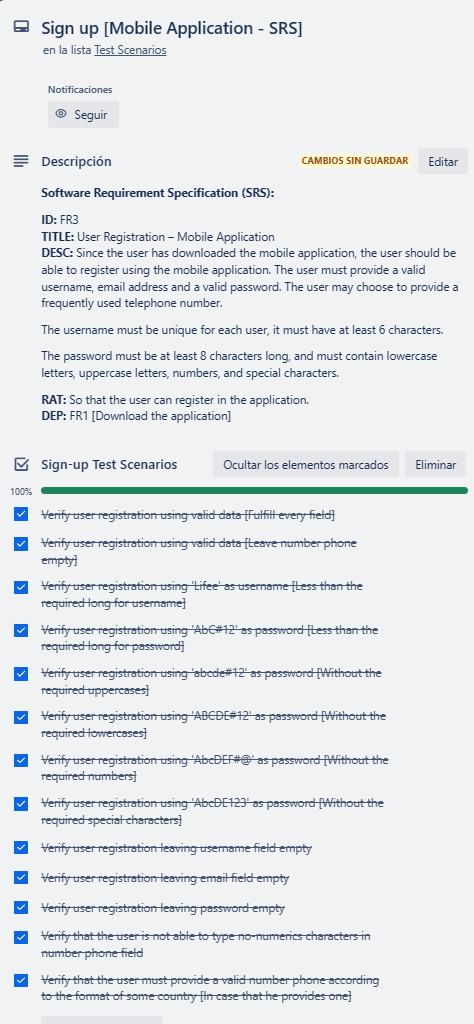
La contraseña debe de poseer al menos 8 caracteres de largo, y debe de poseer letras minúsculas, letras mayúsculas, números y caracteres especiales.

**RAT:** Para que el usuario pueda registrarse en la aplicación.

**DEP:** FR1 [Descargar la aplicación]

**Paso 2:**

En base al paso anterior, escribimos nuestros distintos escenarios de pruebas, tal que así:



**Casos de prueba (Test Cases):**

Es un conjunto de precondiciones, acciones, resultados esperados y poscondiciones desarrollados en base a las condiciones de prueba.

Tienen una serie de campos o pasos:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Título | Test suite | Precondición | Pasos | Resultado esperado | Resultado obtenido | Status | Ambiente |
| Verificar login con e-mail y contraseña válidos | Login | Estar registrado correctamente | 1. Entrar un e-mail valido 2. Entrar una contraseña válida 3. Presionar el botón “Iniciar sesión” | 1. Loguearme exitosamente 2. Ser redirigido a la página de inicio | 1. Logueado exitosamente 2. Redirigdo a la pagina de inicio | Aprobado | Android 13 en Redmi Note 12 Pro 5G |

**Título:** Debe ser un título largo y lo suficientemente autodescriptivo como para saber que se va a testear. Por ejemplo, este título: “Verificar login” es incorrecto, ya que es muy general y poco preciso, ya que no te dice exactamente que o como vas a testear el login, en su lugar sería mejor poner: “Verificar login con e-mail y contraseña válidos”.

**Test suite:** Es el set o conjunto de casos de prueba al que pertenece el caso de prueba, por ejemplo, este caso de prueba pertenece al Test Suite de “Login”.

**Precondición:** Son las condiciones previas requeridas para poder ejecutar el caso de prueba. Por ejemplo, el caso de prueba “Verificar login con e-mail y contraseña válidos” tiene como precondición que el usuario ya se haya registrado correctamente.

**Pasos:** Se describen a detalle los pasos que se darán para ejecutar el caso de prueba. OJO: Es incorrecto enumerar dentro de los pasos cosas excesivamente lógicas e intuitivas, y fuera del alcance del software en sí, por ejemplo, estos pasos no estarían correctos para un caso de prueba de un login:

1. Encender el internet
2. Encender el computador/celular
3. Entrar a Edge
4. Buscar <https://miaplicacion.com>
5. Ir a la opción de login
6. Entrar un e-mail válido
7. Entrar una contraseña válida
8. Presionar el botón “Iniciar sesión”

En su lugar sería mejor esto:

1. Entrar un e-mail válido
2. Entrar una contraseña válida
3. Presionar el botón “Iniciar sesión”

**Resultado esperado:** Es el resultado que debería de obtenerse al ejecutar el caso de prueba.

**Resultado obtenido:** Es el resultado que realmente se obtuvo después de ejecutar el caso de prueba, el cual puede o no ser el mismo resultado que el resultado esperado. NO LLENAMOS ESTE CAMPO HASTA QUE HAYAMOS EJECUTADO EL CASO DE PRUEBA.

**Status:** Es el estado del caso de prueba, tiene estos posibles estados:

* **Nuevo:** El caso de prueba aún no se ha ejecutado.
* **Aprobado:** El resultado obtenido en el caso de prueba es el resultado esperado.
* **Fallido:** El resultado obtenido en el caso de prueba NO es el resultado esperado.
* **Bloqueado/Saltado:** El caso de prueba no puede o pudo ser ejecutado. Debido a que, por ejemplo, alguna precondición para el caso de prueba falló o no paso la prueba, por tanto, todos los casos de prueba dependiente de ese caso de prueba estarán como “Bloqueados” hasta que los desarrolladores arreglen esa funcionalidad y el caso de prueba pase la prueba. Por ejemplo, si el registro no paso la prueba, el caso de prueba de login estará “Bloqueado” ya que el login depende de que el usuario se haya registrado previamente.

**Ambiente:** Es el contexto, environment o dispositivo en el que se ejecuto el caso de prueba, por ejemplo, un caso de prueba se pudo haber ejecutado en Windows 11 Version Home, en el navegador Edge ejecutado en Windows 11 Version Home, o en Android ejecutado en un Xiami Redmi Note 12 Pro 5G. SIEMPRE va en este formato: Sistema Operativo – Hardware o dispositivo - Red

**Tipos de casos de prueba:**

**De alto nivel:** Se escriben cuando estamos diseñando los casos de prueba basados en los requerimientos, es decir, aún no existe una aplicación o página web real. Por tanto, los pasos de estos casos de prueba suelen ser más genéricos y menos específicos que los casos de prueba de bajo nivel. Lo que deja más creatividad al tester de introducir los valores o datos que se le ocurran. Ejemplo:

* **Título:** Verificar un registro con nombre, e-mail, género, contraseña y fecha de nacimiento válidos
* **Precondiciones:** N/A
* **Test suite:** Registro
* **Pasos:**

1. Introducir nombre válido
2. Introducir apellido válido
3. Introducir género válido
4. Introducir e-mail válido
5. Introducir contraseña válida
6. Introducir fecha de nacimiento válida
7. Presionar el botón “Registrar”

* **Resultado esperado:** 
  + Alerta con el mensaje: “Ha sido registrado exitosamente”
  + Redirección a la página de login
* **Resultado obtenido:**
  + Alerta con el mensaje: “Ha sido registrado exitosamente”
  + Redirección a la página de login
* **Status:** Aprobado
* **Ambiente:** Microsoft Edge version 117.043.2554 – Windows 11 Version Home – Wi-Fi 4G LTE

**De bajo nivel:** Se escriben cuando ya existe una aplicación o al menos un aspecto o funcionalidad de la aplicación ya llevada a la realidad para poder realmente ejecutar el caso de prueba. Son extremadamente específicos, por ejemplo:

**Título:** Verificar login con e-mail y contraseña válidos

**Precondiciones:**

1. Estar registrado correctamente

**Test suite:** Login

**Pasos:**

1. Introducir el e-mail [email\_valido@gmail.com]
2. Introducir la contraseña [ContraseñaValida]
3. Presionar el botón “Iniciar sesión”

**Resultado esperado:**

1. Ser loggeado exitosamente
2. Ser redirigido a la página de inicio

**Resultado obtenido:**

1. Loggeado exitosamente
2. Redirigido a la página de inicio

**Status:** Aprobado

**Ambiente:** Google Chrome versión 116.09.432 – Android 14 – Redmi Note 13 Pro - Wifi 5G

**Reportes de defectos:**

Los defectos son imperfecciones o deficiencias del software (bugs, faults or defects en inglés), en el cual no cumple con los requerimientos especificados o tiene un comportamiento inesperado.

Los reportes de defectos se crean una vez ya se han ejecutado las pruebas del software y se encuentra con que alguna parte o funcionalidad no paso la prueba, por lo cual, se procede a crear un reporte de defectos para los desarrolladores, con el siguiente formato:

Section -> Description of where is the defect

* Example: Register -> No error message appears where user leaves password empty

Step to reproduce [must be very specific]:

* Example:

1. Open [www.facebook.com](http://www.facebook.com)
2. Click on the hamburger icon in the upper left corner
3. Scroll down to the bottom of the screen
4. Click on Settings
5. Click on data usage
6. Change data usage to “minimum limit”

Expected result [The same as in the test case]

Actual result [The actual outcome of the test, in the defects reports they are provided, since the tests have been already executed]

Test enviroment [Enviroments where the tests were executed]

Screenshot or video [Depending on the nature of the defect][Must show the whole screen, put a red rectangle around the defect area, and should show clearly the clicks]

Bug priority [Can be critical, high, medium or low] [Critical when the defect is one that affects a too important or key aspect of the application, example: A bug in the voice messages, in a voice application. High when they take part into an important feature but it is not essential for the application, for example a bug in the voice message feature in Facebook. Medium when it is a bug in a “extra” feature of the application, for example the voice message in a payments application. And low when it barely affects the application, for example some typing error, image disablement, etc.

**Tipos de defectos:**

**Funcional:** Son los defectos que afectan una funcionalidad o característica del software. De manera que no funciona o tiene comportamientos inesperados. Por ejemplo, cuando la funcionalidad “Recuperar mi contraseña” no funciona.

**Visual (UI):** Son defectos que afectan a la interfaz de usuario. Por ejemplo: Palabras superpuestas.

**Contenido:** Son defectos que se tratan de contenido dentro de la aplicación que no deberían de estar ahí, por ejemplo, repetirse una frase tanto en el label como el placeholder de un input, o que, en una versión traducida de la aplicación de ingles a árabe, exista alguna palabra en inglés que no esté traducida al árabe.

**Rendimiento:** Son defectos que afectan el rendimiento del software. Por ejemplo, una Http Request o Lazy Loading que toma un tiempo excesivo.

**Sugerencia:** ESTO NO ES UN DEFECTO, sino una manera de sugerir posibles mejoras a la aplicación.

**Técnicas de Testing**

**Técnicas de Caja Negra:**

**Partición de equivalencias:** Se utiliza cuando tenemos que probar una funcionalidad en la que esta involucrada un rango de números (por ejemplo, la edad al registrarse en Facebook, la aplicación de descuento dependiendo del consumo o tipo de usuario que seas, el status de un estudiante dependiendo de la nota que saco, etc.), en la que dividimos los casos de prueba por cada partición que obtengamos, es decir, que si por ejemplo identificamos 3 particiones, obtendremos 3 casos de prueba: 1 por cada partición.

Pueden haber particiones válidas (es decir, particiones en las que los datos son datos válidos y se espera una respuesta funcional o positiva del sistema), y particiones inválidas (es decir, particiones en la que los datos dentro de ella son inválidos y el sistema no debería de permitirlos o procesarlos).

Esto se basa en el principio de que, un objeto de prueba (sistema sujeto a prueba) debería de procesar de la misma manera dos o más elementos o valores de la misma partición, por tanto, solo es necesario un caso de prueba con un solo valor dentro de esa partición.

**Ejemplo:** Funcionalidad de registro en la que la edad admitida es entre 18 y 100 años. La edad se introducirá mediante cajas de texto, poniendo el número correspondiente a tu edad

En este caso identificamos al menos 5 particiones:

Números negativos > Números positivos menores que 18 > Números entre 18 y 100 > Números superiores a 100 > Caracteres como “veinte”

En este caso, solo tenemos una partición válida: La partición entre 18 y 100, todas las demás son inválidas.

Y a continuación entonces, procedemos a crear un caso de uso por cada partición, en la que los datos del test sea un solo valor dentro de la partición.