ACERCA DEL AUTOR

ALEJANDRO JUAN CANOSA FERREIRO

Ingeniero técnico en informática de gestión por la Universidad Nacional de Educación a Distancia. Postgrado en Software Quality Assurance por la Universidad Politécnica de Cataluña. Certificado en Scrum Foundation Professional Certification (SFPC), DevOps Essentials Professional Certification (DEPC), Expert Level Certification en Katalon y a punto de certificarse en ISTQB , Professional Scrum Master e ITIL.

Consultor de QA con más de 10 años de experiencia en el mundo de calidad de software entre España y Colombia y experto en automatización de pruebas de software con herramientas como Selenium, TestComplete, Katalon o Rational Functional Tester de IBM.

1

CONCEPTOS FUNDAMENTALES DE CALIDAD

1.1 QUÉ SON LAS PRUEBAS DE SOFTWARE Y SU IMPORTANCIA EN EL CICLO DE DESARROLLO DE SOFTWARE

Las pruebas de software son fundamentales cuando creamos un software o aplicación, su objetivo es fundamentalmente detectar defectos en el software en cualquier fase de su desarrollo y prevenir la aparición de esos defectos en cualquiera de esas fases.

1.1.1 Objetivos de las pruebas

Los objetivos dependen muchas veces del tipo de prueba, del *contexto* o de la metodología de desarrollo, no tiene los mismos objetivos una prueba funcional que una prueba de regresión o de rendimiento o incluso cuando se hacen las pruebas depende mucho de si es una metodología en cascada o una metodología ágil, pero a modo general y teniendo en cuenta todos los distintos tipos de prueba que hay, que hablaremos más adelante de ellas podemos indicar los siguientes:

- *Prevenir la aparición de defectos.*
- *Verificar que se cumplen los* ***requerimientos.***
- *Generar confianza en la calidad del producto que se está desarrollando.*
- *Encontrar defectos para solucionarlos y que no llegue a* ***producción.***
- *Sirve para obtener información muy importante para la toma de decisiones.*
- *Cumplir estándares.*

A continuación, paso a describir algunos conceptos que más adelante se explicaran más en profundidad pero que ahora quiero explicar un poco.

Cuando hablo de contexto me refiero a que hay veces que las pruebas tienen objetivos y se realizan en momentos muy diferentes, por ejemplo, una prueba unitaria se realiza para probar todo el código mientras que una prueba de aceptación se realiza para verificar que el producto entregado cumple con las exigencias que solicitó el cliente.

Los requerimientos son las necesidades que el cliente que nos solicita el software necesita y es fundamental que se cumpla o puede que el cliente no quiera el producto y tengamos que volver a desarrollarlo desde la fase inicial lo que conllevaría retraso en la entrega y pérdidas enormes, por eso es tan importante la captura de requerimientos lo más fiel posible a las necesidades del cliente, pero eso se verá más adelante.

Producción es el ambiente donde el cliente instala el software y donde se hará el uso continuado por parte del cliente, ya lo veremos, pero en desarrollo y calidad de software hay varios ambientes y el último y más importante es el de producción.

1.1.2 Importancia de las pruebas de software

Son muy importantes por muchas razones, pero las más importantes serian:

- Ayuda a reducir el riesgo de un error en el entorno real lo que afectaría a la imagen de la empresa, su credibilidad o incluso su facturación anual o mensual; imaginemos que El Corte Ingles que depende su supervivencia de la época de Navidad y del inicio del cole tuviera un fallo en su sistema central de cobro y sus terminales no pudieran cobrar durante 24 horas el día antes del comienzo de la clases la pérdida económica y el daño a su imagen, ya muy deteriorada por su crisis actual, sería brutal, afianzaría la imagen que se tiene que se ha quedado obsoleto comparado con Amazon, Mango o Inditex empresas además que utilizan las últimas técnicas y tecnologías de la calidad de software.

- Puede ser requerido por entes reguladores, algo que les pasa a los bancos continuamente como el BBVA, el Banco Santander, la Caixa, etc., no cumplir con los estándares del gobierno y la ley puede acarrearles pérdidas económicas, multas y hasta una pedida en su imagen pública.
- Las pruebas permiten que un proyecto se realice adecuadamente, que se cumpla con los requerimientos del cliente y que no haya perdidas no esperadas además que cuanta más calidad tiene un producto mayor es la satisfacción del cliente y por lo tanto se consigue más lealtad en la compra, ¿creéis que si Apple no imprimiera esa calidad en sus componentes y en su diseño habría gente tan fan de su marca?, yo creo que no y un cliente contento paga más por algo y habla a otros de tu marca.

1.1.3 Impacto de no realizar pruebas de software

Puede parecer que las pruebas que se realizan durante el proceso de desarrollo de un software no son importantes, pero afectan de manera determinante tanto en el tiempo de entrega de ese software como en las repercusiones que tienen para el cliente que va a utilizar ese software.

Las consecuencias más importantes suelen ocurrir en el ambiente real, es decir, cuando el software ya está en producción y se está utilizando por los usuarios de la empresa que solicitó el software o aplicación y algunas de ellas serian.

- *Pérdida económica de la empresa.*
- *Mala reputación de la empresa.*
- *Pérdida de credibilidad de la empresa.*
- *Pérdida de tiempo.*
- *Pueden ocurrir catástrofes.*
- *Lesiones.*
- *Muertes.*
- *No cumplir con estándares obligatorios del gobierno.*
- *Perdida de satisfacción del cliente.*
- *Que la aplicación no sea lo que el cliente esperaba.*

1.2 ERROR, DEFECTO Y FALLO

Vamos a diferenciar qué es error, qué es un defecto y qué es un fallo y hablare de varios ejemplos de cómo un defecto puede producir un fallo que genera pérdida económica y mala reputación a una empresa.

1.2.1 Error

El error es una equivocación de una persona, realmente es un fallo humano y ocurre porque no somos máquinas y siempre podemos cometer un error.

1.2.2 Defecto

El defecto es una consecuencia del error humano que puede ser de un desarrollador o algunas veces del analista de requerimientos o del analista funcional.

1.2.3 Fallo

El fallo es una consecuencia del defecto ocurrido en el proceso de desarrollo del software y digo proceso y no solo la codificación del software porque puede ocurrir al tomar mal los requerimientos del usuario o al diseñar mal una funcionalidad por eso es muy importante probar todas las fases de desarrollo del software.

1.2.4 Ejemplos de efectos reales de un defecto

Un ejemplo de un fallo podría ser el que al leer el código de barras de un producto en una tienda de zapatos el software si no lee bien el código de barras no pida su repetición, si no que ponga un precio por defecto esto generaría pérdidas a la zapatería importantes.

Otro fallo que demuestra lo importante que es hacer pruebas de rendimiento o de seguridad es las caídas que sufre de vez en cuando WhatsApp, Instagram o Facebook, genera una alarma a nivel mundial, baja satisfacción del cliente y hasta seguramente perdida de usuarios y de imagen sobre todo si ocurren a menudo, también muchas personas se han pasado a Telegram o han dejado su cuenta en Instagram o Facebook por la sensación de falta de privacidad lo que demuestra que las pruebas de seguridad y rendimiento son importantes y digo seguridad porque cumplir con **LOPD**(Ley orgánica de protección de datos española) y **RGPD** (Ley general de protección de datos europea) forma parte de las pruebas de seguridad aunque muchas empresas americanas no les hagan caso por un intento de espiar el consumo de sus clientes para ofrecerles lo que realmente les interesa.

1.3 LOS 7 FUNDAMENTOS DEL TESTING O PRUEBAS DE SOFTWARE

La calidad de software, aseguramiento de calidad o QA en inglés Quality Assurement tiene unos principios, no son 10 como en las tablas de Moisés, ja, ja, ja., pero son 7 y paso a enunciarlos y a explicarlos.

1. **Las pruebas muestran la presencia de defectos no su ausencia o no existencia**

   Esto significa que cuando realizamos unas pruebas de cualquier tipo o de cualquier nivel y no encontramos defectos no significa que no tenga, sino que estas pruebas no los detectaron, puede ser que no se han hecho las pruebas con suficiente profundidad, que han sido muy rápidas porque generalmente siempre se encuentran defectos y es muy raro que no encontraron ninguno.

2. **Las pruebas exhaustivas son imposibles**

   Es imposible probar un software completamente por dos razones, la primera porque si utilizas una metodología ágil como SCRUM hay un tiempo para hacer pruebas y no puedes extenderte semanas porque un sprint suele tardar como mucho eso y dos porque ningún analista es capaz de ver todas las posibilidades que pueden ocurrir, las combinaciones son muchísimas y por lo tanto también los escenarios de prueba.

   No te preocupes si no sabes qué es un escenario de prueba ni qué es analista, son cosas que más adelante te explicare.

3. **Las pruebas tempranas ahorran mucho tiempo y dinero**

   Las pruebas deberían empezar desde la captura de requerimientos con el cliente (es la primera fase en cualquier metodología de desarrollo ágil o clásica).

4. **Los defectos se agrupan**

   Si se encuentran defectos en una funcionalidad o una clase seguro que habrá más defectos porque los defectos suelen generar más defectos y hay mucha más probabilidad de en una parte que ya haya se encuentren más; de hecho, una buena práctica es tener un historial de defectos por funcionalidad y por versión del software para poder tener especial cuidado en esas partes tanto con la codificación del código como con sus correspondientes pruebas.

5. **Los casos de prueba deben actualizarse**

   Una de las creencias que se tiene cuando se realizan los casos de prueba (por ahora solo quiero que sepas que existen, en breve te hablare de ellos, su función, técnicas para realizarlos y que forman parte de un proceso de pruebas del que también te hablare);como te comentaba los casos de prueba son un ente vivo al igual que las aplicaciones que se construyen a esa aplicación se le añaden nuevas funciones, se eliminan otras y se actualizan otras por lo tanto los casos de prueba que sirven para probar esas funcionalidades deben ser eliminados o marcados como obsoletos, actualizados y crear nuevos casos de prueba para probar esas funcionalidades para que el desarrollo cumpla con una mínima seguridad de que no fallara en producción.

6. **Las pruebas dependen del contexto**

   Las pruebas no son iguales cuando están probando el código que cuando están probando su funcionalidad o su rendimiento o tampoco si se realizan en entorno de desarrollo, que en **UAT** u otros entornos que existan; hablare sobre los entornos que existen y cuales bajo mi experiencia son fundamentales para tener la máxima calidad posible.

7. **La falacia de la ausencia de errores**

   Esto va dirigido a los jefes de proyectos, CEO `s o gerentes si ustedes ven un informe sin defectos de ningún tipo, les están tomando el pelo, siempre hay defectos, bien en el código, bien en el rendimiento o en su funcionalidad y si no hay defectos es solo porque no hubo tiempo suficiente para probar y se probaron las funcionalidades más importante, está claro que no todos los defectos son igual de críticos y hay que intentar que una versión salga a producción con los defectos menos críticos pero cierto es que un defecto poco crítico con el tiempo puede ser un gran defecto como esa gotera que no hace nada hasta que pasan meses y de repente va a más o con el tiempo esa gotera termina dañando algo importante; no poder encontrar todos los defectos ni solucionar todos los defectos no significa que poco a poco vayan siendo eliminados y entiendan un concepto, siempre se puede mejorar.

1.4 ¿QUÉ SON LAS PRUEBAS ÁGILES?

Hoy en día en un mundo globalizado, donde continuamente salen nuevas aplicaciones e ideas no sirve solamente con tener una buena idea hay que crearla rápidamente y con una calidad que cumpla con las necesidades del mercado, por eso las metodologías ágiles son tan importantes.

La filosofía ágil consiste en utilizar la integración continua, las entregas continuas, la colaboración entre equipos y un ***feedback*** continuado para que el equipo se empape de esa cultura, esa forma de trabajar para lanzar software continuamente, pero de calidad.

En esta cultura ágil y en esta forma de trabajar ágil también las pruebas deben ser ágiles, ¿qué significa esto?, que son las pruebas ágiles.

Las pruebas ágiles son una metodología de trabajo que tiene dos pilares fundamentales: ***feedback*** continúo y generar software de calidad.

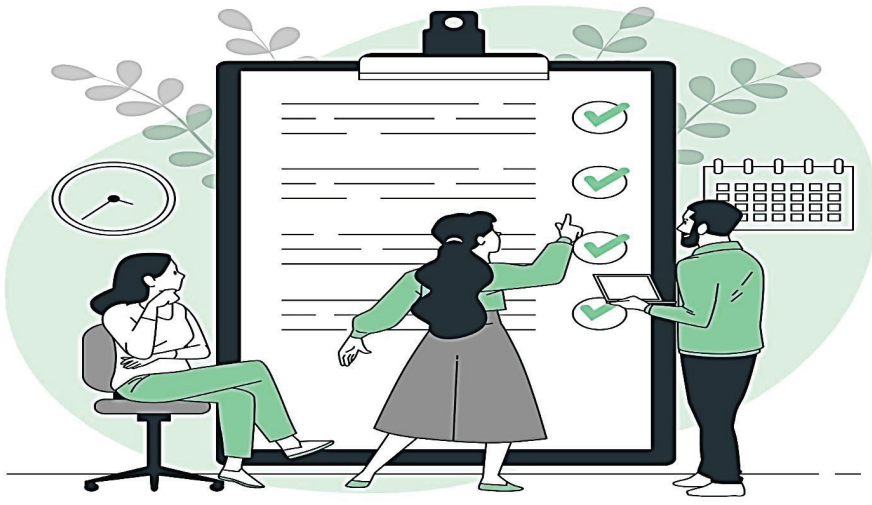
Estas pruebas ágiles se basan en el concepto ágil de desarrollo iterativo e incremental.

A diferencia de otras metodologías de pruebas las pruebas ágiles suceden a la vez que se está desarrollando el software, en cada fase del proceso de desarrollo de software los testers encuentran nuevos errores lo que permite que se puedan ir solucionando y no lleguen a fases más avanzadas donde el coste de resolverlo puede ser mucho más alto.

Las continuas iteraciones que ocurren dentro de un proyecto ágil incluyen comentarios de, **testers**, desarrolladores y usuarios finales lo que ayuda a una mejora continua del software.

Para implantar correctamente las pruebas ágiles no solo sirve con probar en cada fase del desarrollo también crear un equipo multidisciplinar donde los principios de agilidad sean aplicados, hay un ***manifiesto ágil*** que engloba los principios de cómo ser una organización ágil no solo a la hora de desarrollar un software si no en la propia estructura y organización de la empresa y en su manera de comunicarse y trabajar, de estos principios de agilidad hablaré en próximos temas por ahora solo quiero que se entienda que son las ***pruebas ágiles*** y que hay un ***manifiesto ágil*** que aplican muchas empresas en su manera de trabajar cada día.

1.5 QUÉ SON LOS CASOS DE PRUEBA Y EJEMPLO DE UNO



1.5.1 Qué es un caso de prueba

Los casos de prueba son un conjunto de acciones ejecutadas para comprobar una funcionalidad de un software.

Un caso de prueba está formado por varios componentes:

- Los pasos del caso de prueba: los pasos que hay que realizar para ejecutar el caso de prueba.
- Los datos del caso de prueba: los datos que se utilizan para poder realizar la prueba.

- Pueden tener precondiciones y postcondiciones: no siempre son necesarios y cuando es una precondición se tiene que cumplir antes de ejecutar la prueba y cuando es una postcondición después de ejecutar la prueba.

Los casos de prueba forman parte de un escenario de prueba y pueden probar cualquier tipo de software o aplicación.

Los casos de prueba pueden ejecutarse de manera manual, automatizada o utilizando un software de gestión de pruebas como **XRAY**, un componente de **JIRA.**

Algunas veces hay dudas con dos conceptos que pueden parecer iguales, pero no lo son que son el caso de prueba y el escenario de prueba, el caso de prueba es un conjunto de pasos que se tienen que realizar para probar una funcionalidad y ver si funciona como se espera y un escenario de prueba es la funcionalidad que se quiere probar, quizás con un ejemplo se verá mejor.

Probar la funcionalidad de iniciar sesión en una aplicación sería el escenario de prueba y el caso de prueba serian todos los pasos que hay que seguir, junto a los datos necesarios y las precondiciones y postcondiciones si los tuviera.

1.5.1 Cómo escribir un caso de prueba

Un caso de prueba tiene los siguientes campos, indicare cuales son y luego crearemos un caso de prueba para que podáis realizarlo vosotros desde vuestro ordenador en una página de mi propiedad.

- **ID del caso de prueba**: este identificar debe ser único y sirve para identificarlo de manera única dentro de un grupo de casos de prueba.
- **Título del caso de prueba**: un nombre que identifique el caso de prueba de una manera más clara.
- **Descripción del caso de prueba**: se describe la funcionalidad que se va a probar y su objetivo.
- **Condiciones previas**: son condiciones que se deben cumplir antes de realizar el caso de prueba.
- **Pasos de la prueba**: son los pasos que hay que seguir para poder realizar la prueba.
- **Resultado esperado**: es el resultado que se espera después de cada paso realizado.
- **Condición posterior**: es la condición que debe cumplir después de realizar la prueba.

Con respecto a escribir un caso de prueba las recomendaciones son:

- Nunca utilizar primeras personas.
- Escribe siempre como si la persona no tuviera ni idea de la aplicación.
- Cada paso debe tener un resultado esperado.
- Cada dato necesario comprobar antes para asegurarse que es correcto y no ha cambiado.
- Si se puede, adjuntar unas imágenes con las pantallas para que sea más fácil de identificar la funcionalidad que se va a probar y cada uno de los pasos.

Todo esto lo comento porque he estado en proyectos en las que mi función era automatizar pruebas funcionales y es muy complicado cuando no tienes tiempo y ni conoces la aplicación ni ninguno de los funcionales tiene tiempo de explicarte la funcionalidad poder probar o automatizar un caso de prueba por eso es fundamental que haya miembros del equipo que validen los casos de prueba creados y que sea gente que no conozca la aplicación para saber hasta qué punto alguien que no conoce la aplicación es capaz de realizar una prueba.

El ser humano por naturaleza tiende a no ser muy generoso con su conocimiento sobre todo en puestos en que tienen miedo de ser prescindible pero no puede ser que los miedos generen una calidad ínfima sobre todo en algo tan importante para probar una funcionalidad como un caso de prueba.

1.5.2 Ejecución de un caso de prueba en un entorno real

Vamos a ejecutar el caso de prueba iniciar sesión en una tienda virtual que se llama *www.pruebasqafenix.com*, es una aplicación donde podremos realizar todo tipo de pruebas y que utilizaremos en varias prácticas de este libro, la ejecución del caso de prueba tendrá más campos que los indicados en el caso de prueba y se puede ver en la siguiente tabla:

ID= 1

Titulo= Inicio de sesión de un cliente con cuenta en la tienda virtual.

Descripción= Comprobar que un cliente de la tienda virtual pruebasqafenix.com puede acceder correctamente a su página de cliente registrado.

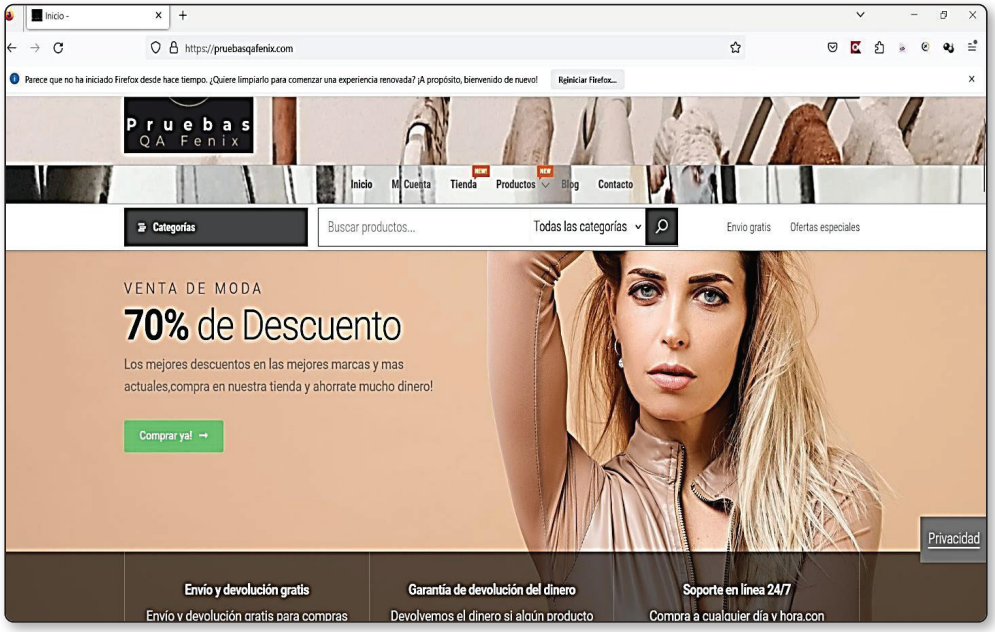
Condiciones previas= La página tiene que estar online y con un tiempo de respuesta inferior a 10 segundos y tener un usuario con cuenta en la página.

Pasos	Datos	Resultado esperado	Resultado real	Estado
Abrir un navegador				OK
Escribir en la barra de navegación la url de la tienda virtual	www.pruebasqafenix.com	Se carga correctamente y en menos de 10 segundos	Se carga correctamente y en menos de 10 segundos	OK
En el menú principal hacer clic en **Mi Cuenta**		Se carga correctamente y se muestra dos formularios el de acceder y registrarse	Se carga correctamente y se muestra dos formularios el de acceder y registrarse	OK
En el campo usuario escribir el usuario	Juan. Sánchez	Se muestra el texto en el campo usuario	Se muestra el texto en el campo usuario	OK
En el campo contraseña escribir la contraseña	Deportivo 2023	Se muestra la contraseña en asteriscos en el campo contraseña con tantos asteriscos como caracteres	Se muestra la contraseña en asteriscos en el campo contraseña con tantos asteriscos como caracteres	OK
Hacer clic con el ratón en el botón acceder		Accede a la página de inicio de mi cuenta	Accede a la página de inicio de mi cuenta	OK

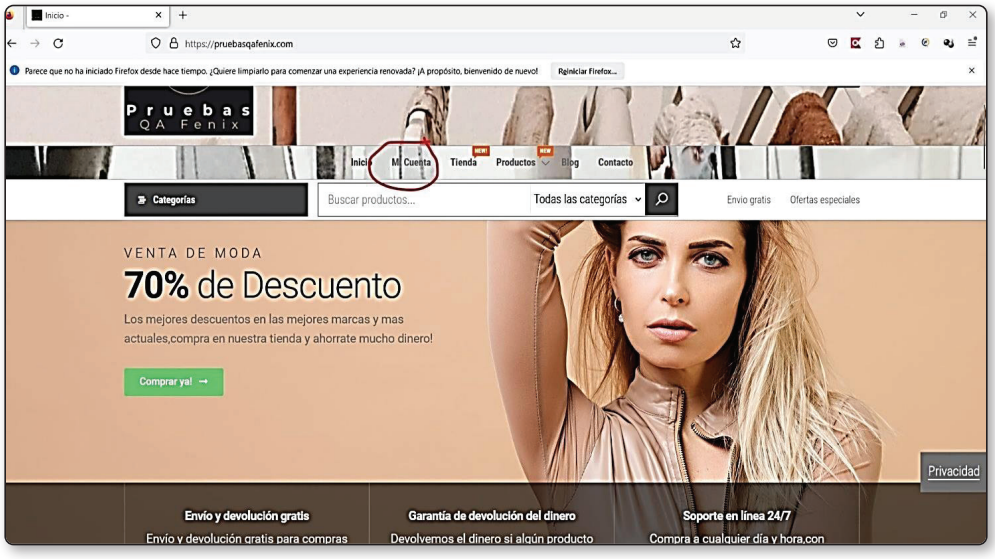
Una buena postcondición sería cerrar la sesión y el navegador para que volviera estar el escenario en su estado inicial, esto se hace mucho en entornos donde se automatiza la prueba.

Un buen caso de prueba debería tener los pasos grabados en un video o imágenes porque así se vería mucho más claramente los pasos, por eso lo vamos a hacer en este ejemplo práctico.

- **PASO 1.**

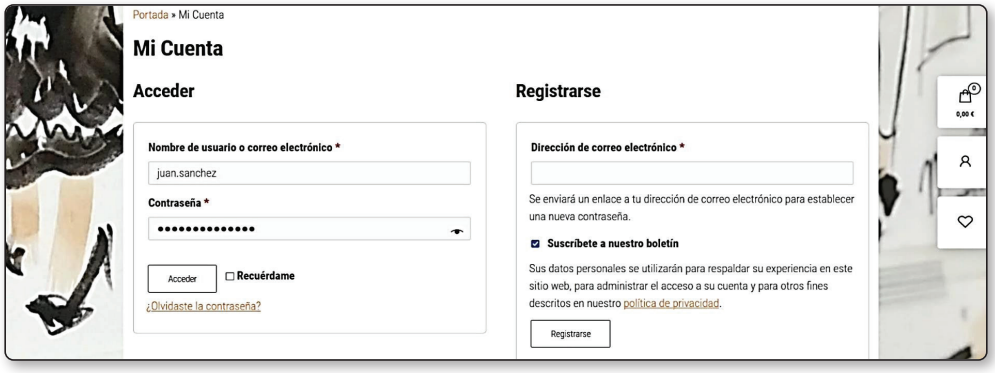

Inicio -
https://pruebasqafenix.com
Parece que no ha iniciado Firefox desde hace tiempo. ¿Quiere limpiarlo para comenzar una experiencia renovada? ¡A propósito, bienvenido de nuevo!
Reiniciar Firefox...
Pruebas QA Fenix
Inicio
Mi Cuenta
Tienda
Productos
Blog
Contacto
Categorías
Buscar productos...
Todas las categorías
Envío gratis
Ofertas especiales
VENTA DE MODA
70% de Descuento
Los mejores descuentos en las mejores marcas y mas actuales,compra en nuestra tienda y ahorrate mucho dinero!
Comprar ya!
Privacidad
Envío y devolución gratis
Envío y devolución gratis para compras
Garantía de devolución del dinero
Devolvemos el dinero si algún producto
Soporte en línea 24/7
Compra a cualquier día y hora.con

- **PASO 2.**


Inicio -
https://pruebasqafenix.com
Parece que no ha iniciado Firefox desde hace tiempo. ¿Quiere limpiarlo para comenzar una experiencia renovada? ¡A propósito, bienvenido de nuevo!
Reiniciar Firefox...
Pruebas QA Fenix
Inicio
Mi Cuenta
Tienda
Productos
Blog
Contacto
Categorías
Buscar productos...
Todas las categorías
Envío gratis
Ofertas especiales
VENTA DE MODA
70% de Descuento
Los mejores descuentos en las mejores marcas y mas actuales,compra en nuestra tienda y ahorrate mucho dinero!
Comprar ya!
Privacidad
Envío y devolución gratis
Envío y devolución gratis para compras
Garantía de devolución del dinero
Devolvemos el dinero si algún producto
Soporte en línea 24/7
Compra a cualquier día y hora.con

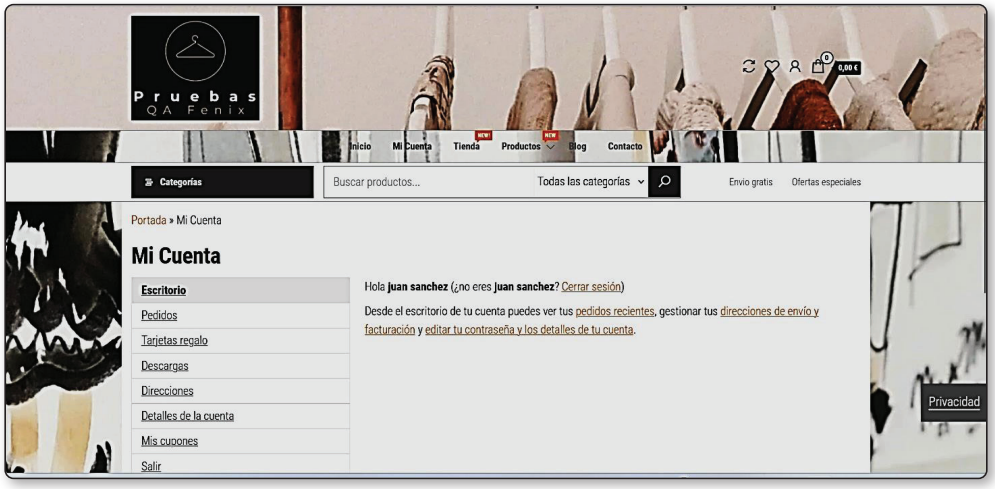
- **PASO 3.**

- **PASO 4.**



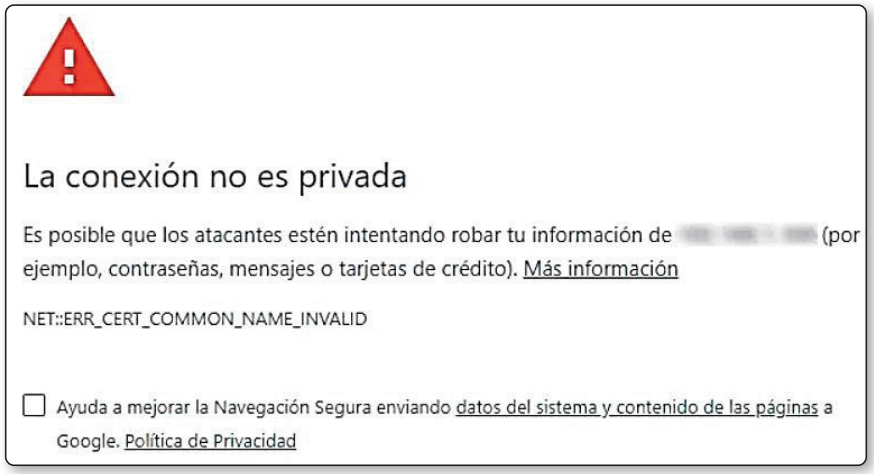
- **PASO 5.**

- **PASO 6.**



1.6 QUÉ SON LOS FALLOS Y EJEMPLO DE UNO

Un fallo realmente es cuando en producción ocurre algo inesperado, por ejemplo, intentamos acceder a nuestra tienda virtual favorita y nos da un error de que el certificado no es seguro, algo como lo de abajo.



Este error ocurre cuando el certificado digital de la página no concuerda con el dominio.

Pero estamos en un libro de **QA** y de **SCRUM** y lo que nos interesa es un fallo que tiene que ver más con el proceso de desarrollo.

Supongamos que en los requerimientos sacados de las reuniones con el cliente nos pide que los clientes de la tienda virtual que estamos construyendo al iniciar sesión accedan la página de inicio de su cuenta pero cuando estamos en producción los clientes nos indican que al acceder están siendo redirigidos al carrito de la tienda; este tipo de error es un error funcional y genera una molestia sería a los clientes porque no permite seguir el flujo normal de una historia tan importante como la de iniciar sesión ya que es algo que todos los usuarios verán en cuanto inicien sesión.

1.7 ESTRUCTURA DE UN REPORTE DE UN BUG O FALLO

Vamos a hablar de la estructura que tendría el reporte de un error, luego veremos cómo hacerlo en una herramienta como Jira y posteriormente el flujo que seguiría ese bug en un entorno real y cómo se solucionaría.

Las partes de un defecto serian:

- *Identificador= Un identificador único.*
- *Título= Un título que resuma el error.*
- *Descripción= Un pequeño resumen del error.*
- *Fecha.*
- *Autor= Quien encontró el defecto y lo creo en la herramienta.*
- *Identificación del elemento que pruebas= El nombre de la historia que estas probando.*
- *Fase del ciclo de desarrollo (sprint).*
- *Detalles para reproducirlo= Suele ser o pasos o un video con los pasos.*
- *Imagen o video del defecto= Una imagen en formato. jpf o un video .mp4.*
- *Resultado actual= Es lo que está ocurriendo.*
- *Resultado esperado= Es lo que debería ocurrir.*
- *Impacto= Puede ser crítico, severo, alto, normal o bajo.*
- *Prioridad= Muy alta, alta, media, baja, muy baja.*
- *Estado= Si está solucionado, si está a la espera de estar solucionado o si está a la espera de ser probado.*

1.8 JIRA COMO HERRAMIENTA PARA GESTIONAR UN PROYECTO ÁGIL

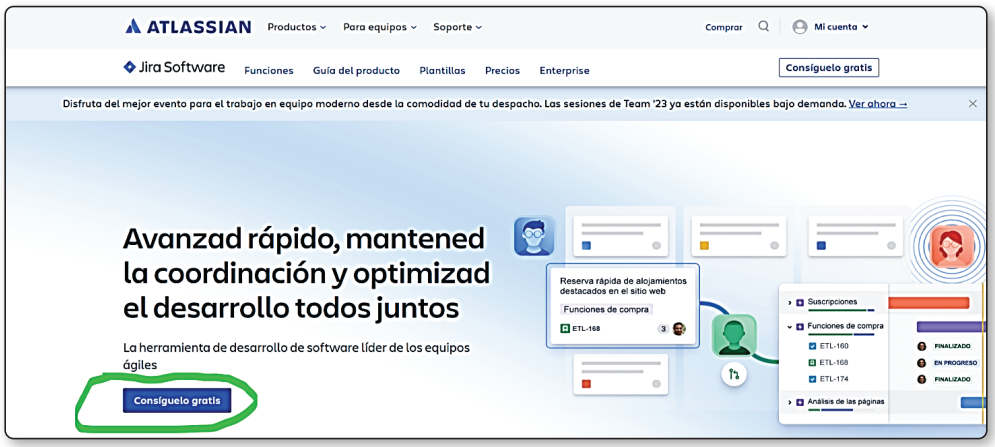
Quiero hacer una introducción a Jira porque es la herramienta que vamos a utilizar para mostrar la metodología **SCRUM**.

JIRA es la herramienta más utilizada a día de hoy la para gestión de proyectos ágiles que utilizan metodologías ágiles como **Kanban**, **XP** o **SCRUM,** pero originalmente fue diseñada para rastrear y gestionar errores en el desarrollo de un proyecto de software.

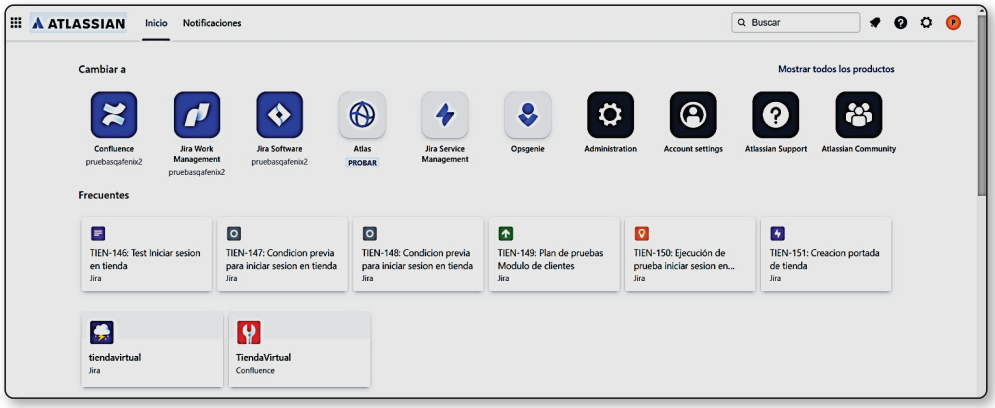
JIRA tiene 3 núcleos fundamentales:

- **Jira Software**= Es para gestionar proyectos de creación de software sobre todo metodología SCRUM y Kanban.
- **Confluence**= Es para la gestión documental y se suele utilizar para documentación relacionada a los proyectos que existen en JIRA.
- **Jira Service Management**= Es para contacto entre clientes y empleados en grandes empresas para que sea ágil y rápido sobre todo cuando son quejas de clientes o correos importantes entre clientes.

Para nuestra práctica en la última sección de este tema tendremos que crear una cuenta en JIRA solo necesitaras un correo dado que tiene plan gratuito, pulsando en ese botón con líneas verdes podrás darte de alta en la plataforma, de todas maneras indicare todos los pasos para darse de alta, crear un proyecto de SCRUM, crear requerimientos, historias de usuarios, errores, tareas y todo lo que necesitemos.



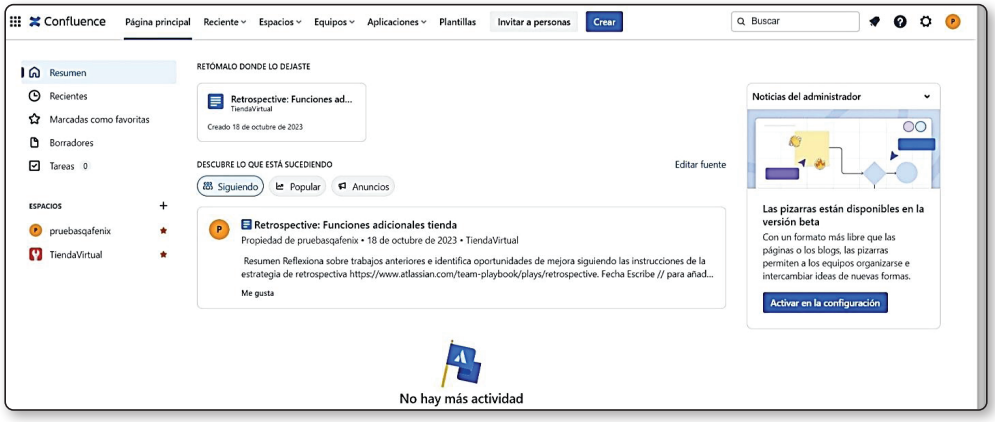
Abajo se puede ver algunos de los módulos que tiene JIRA.



Abajo se puede ver **JIRA Software** y algunas plantillas para crear distintos tipos de proyectos.

Se puede solicitar probar el módulo **Jira Service Management.**

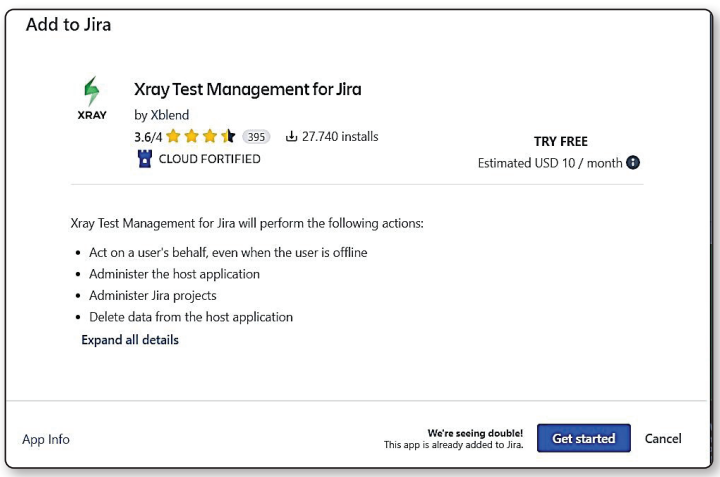
Y por último podemos ver el módulo **Confluence** que como comentamos se utiliza mucho para gestionar todos los documentos asociados a un proyecto, capacitación de un proyecto, acuerdos a los que se llega, etc.



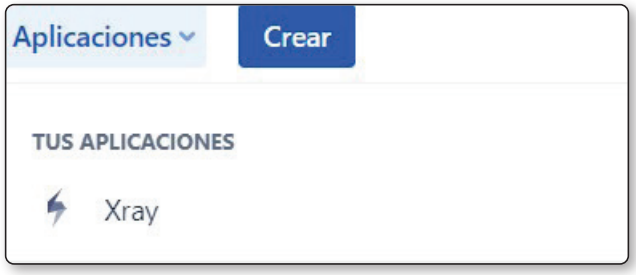
1.9 CICLO DE VIDA DE UN BUG EN JIRA

Jira como comentamos, se creó de inicio para gestionar errores pero con el tiempo se transformó en una plataforma para gestionar todo tipo de proyectos y además tiene un montón de plugins para añadir funcionalidades a la plataforma desde automatizar pruebas, a gestionar pruebas, enlazar con herramientas como **Slack** todo eso se consigue gracias a su módulo de plugins y el que nos interesa ahora es **XRay**, un plugin que nos permitirá crear planes de prueba, ejecutar pruebas, crear pruebas y reportar defectos y hacer un seguimiento de los defectos.

Desde aplicaciones podemos instalar en **XRay** a **Jira.**

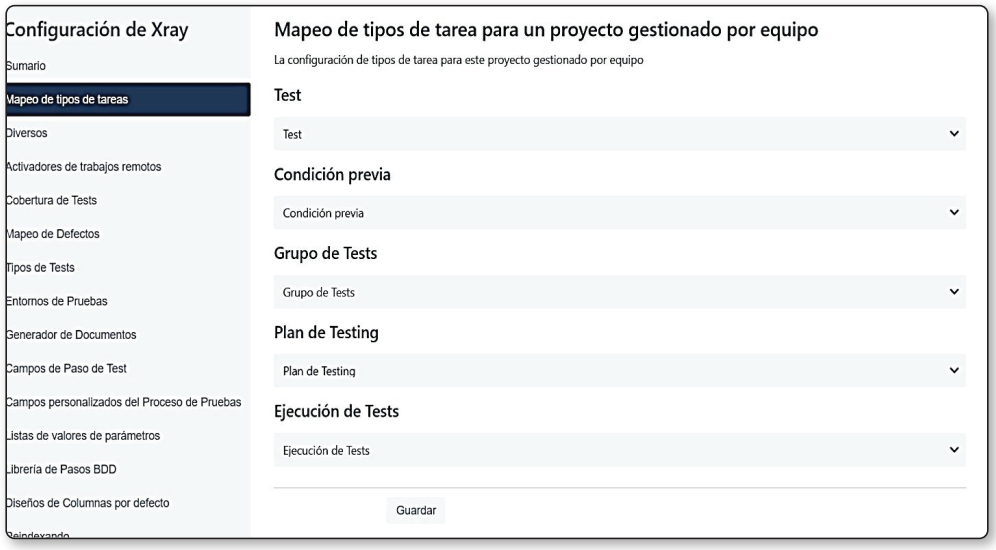


Pulsando en *Get started* podemos probarlo y ya luego por usuario pagar 10 dólares mes; al instalarlo en Aplicaciones nos aparecerá el icono de **XRay.**

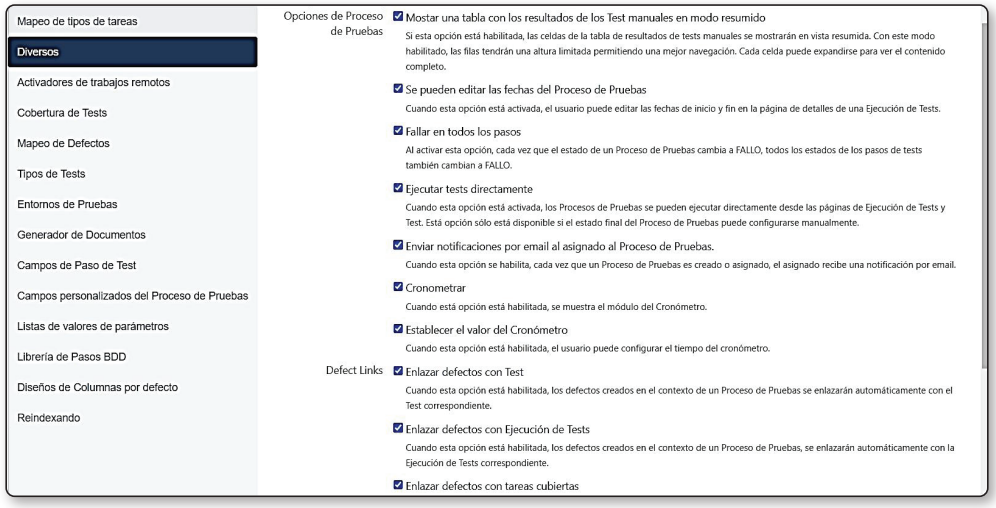


Al pulsar en **XRay** podemos acceder a la pantalla general del plugin para configurarlo, yo ya lo tengo configurado y os lo muestro para que veáis como debería estar.

Las tareas de tipo **QA** en un proyecto serían las que se muestran en la imagen.

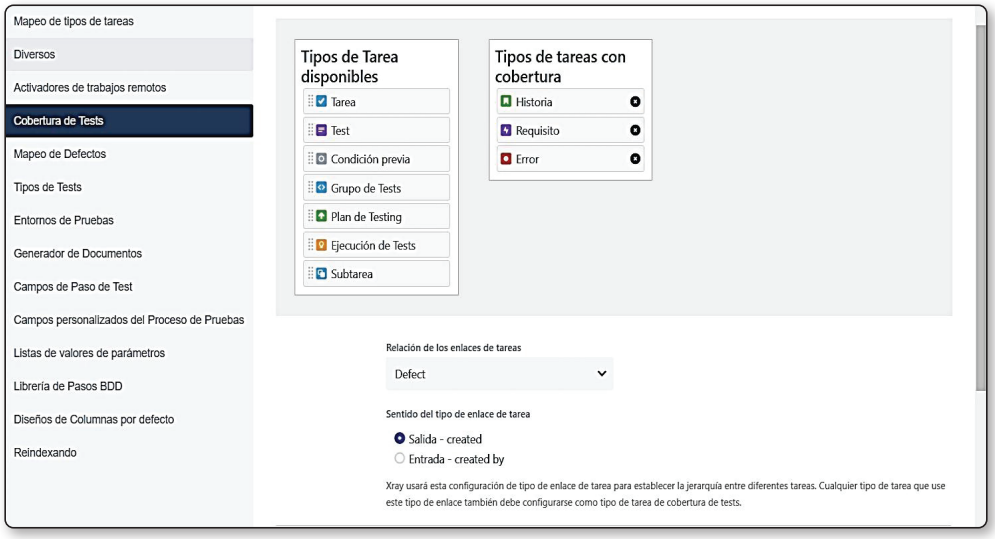


La configuración general de **XRay** es la siguiente.



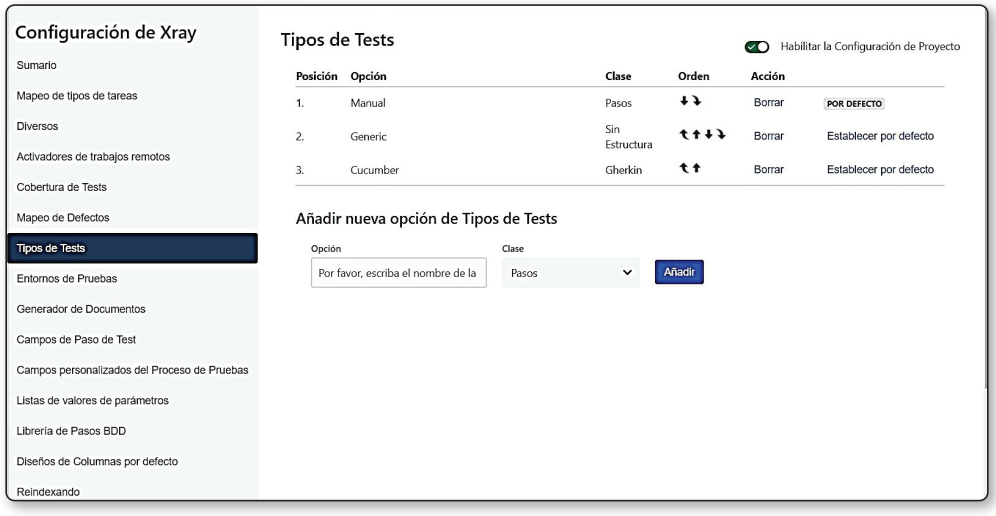
XRay tiene la opción de que se active una ejecución de manera remota desde herramientas **CI/CD** como **Jenkins**, **Bamboo**, etc., pero necesitas la licencia **Enteprise.**

Vamos a suponer que solo vamos a realizar pruebas a historias de usuario, requerimientos y errores porque es lo más importante, los errores porque habrá que verificar que se han solucionado y los requerimientos porque hay que verificar que son los que realmente quiere el cliente porque el costo de crear funcionalidades de una forma distinta a lo que busca el **stakeholder** o la gente involucrada en el proyecto es muy alta pero puedes realizar pruebas de tareas y subtareas de una historia de usuario si así lo queréis.

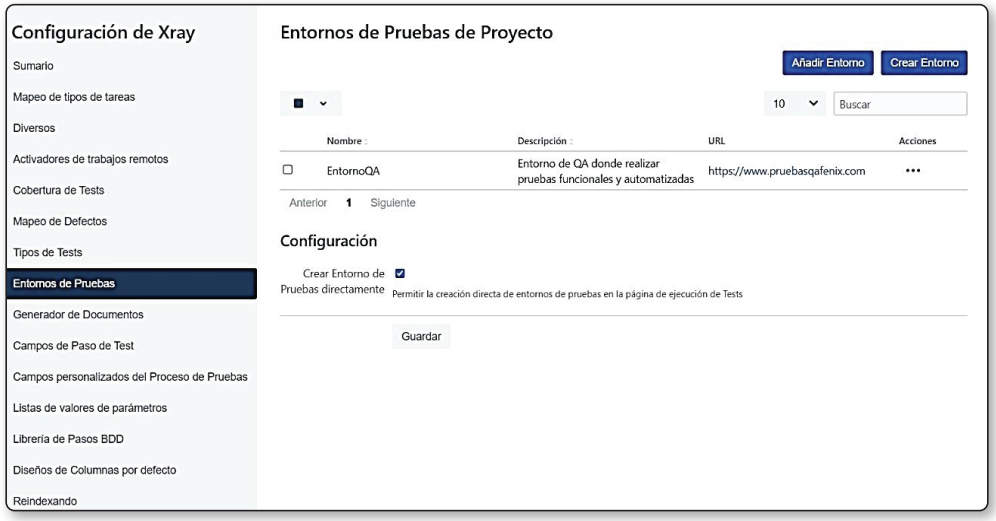


En un proyecto puede haber distintos defectos, puede llamarse bug cuando se encuentra en el código, defecto cuando es en la historia de usuario (diseño de la funcionalidad) u obtención de requerimientos en el momento en que se habla con el cliente y se captura sus necesidades o un fallo que podría ser cuando ocurre en producción, pero en este proyecto que he creado solo hay un tipo de defecto, pero como digo podéis diferenciarlos.

Los tipos de pruebas que podemos tener en **XRay** son 3, manuales, genéricas y por **Cucumber** que es un framework **BDD** (1).

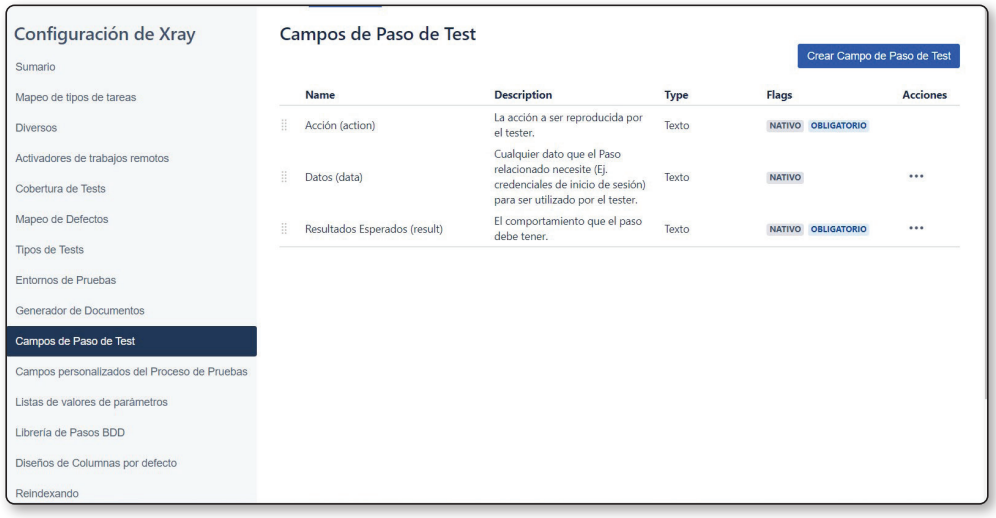


Los entornos de prueba en un proyecto suelen ser el de **QA,** desarrollo, **UAT**, en mi proyecto he puesto solo uno, pero quizás más adelante necesitemos dos más.



(1) Significa Behavior Driven Development (BDD), o desarrollo orientado al comportamiento, es una metodología ágil de desarrollo software en la que una aplicación se documenta y diseña en función del comportamiento que los usuarios esperan o desean experimentar al interactuar con ella.

La última parte importante en la configuración es la que nos permite configurar las columnas de un paso de un caso de prueba en general suelen tener solo las 3 columnas siguientes.



El resto de las secciones por ahora no son importantes; ya tenemos configurado una funcionalidad dentro de **JIRA** para poder gestionar las pruebas y nos vendrá bien para la práctica de este primer tema donde ya hablamos de un ciclo completo de un bug dentro del proceso de desarrollo de una tienda virtual.

En la sección siguiente hablaremos de qué es un caso de prueba y las técnicas para diseñar un caso de prueba.

1.10 TÉCNICAS DE DISEÑO DE CASOS DE PRUEBA

Lo primero que hay que tener en cuenta es qué es un caso de prueba y para qué sirve.

1.10.1 Qué es un caso de prueba, para qué sirve y partes

Los casos de prueba son un conjunto de pasos que se utilizan para validar una característica determinada que puede ser funcional o no funcional y sirve para:

- *Validar que esa característica funciona como se espera.*
- *Ofrece la seguridad al cliente que la aplicación se está construyendo adecuadamente.*
- *Algunos sectores como los de sanidad o banca deben cumplir estándares de calidad estrictos y es obligatorio pasarlos.*

Un caso de prueba tiene las siguientes partes:

1. *ID= Idéntica al caso de prueba de manera única.*
2. *Titulo= Una descripción corta de lo que se va a probar, por ejemplo, inicio de sesión de usuario.*
3. *Precondición= Alguna condición que debe cumplir para poder realizar el caso de prueba.*
4. *Datos= Los datos necesarios para realizar la prueba.*
5. *Pasos de la prueba= Los pasos para realizar la prueba.*
6. *Resultado esperado= El resultado esperado para cada paso de prueba.*

1.10.2 Técnicas para realizar casos de prueba

Las técnicas que debemos utilizar dependen de varios factores:

- *La complejidad del sistema o del componente.*
- *De estándares regulatorios.*
- *Requerimientos de usuarios.*
- *Niveles de riesgo.*
- *Documentación disponible.*
- *Del conocimiento y habilidades del tester.*

Al final las 3 técnicas que tenemos para diseñar casos de prueba son:

1. *Técnicas de prueba de caja negra.*
   Basadas en requerimientos funcionales y no funcionales.
2. *Técnicas de prueba de caja blanca.*
   Basadas en el código fuente de la aplicación y su arquitectura.
3. *Técnicas de prueba basadas en la experiencia.*
   Están basadas en la experiencia de los testers y analistas de calidad.

1.10.2.1 TÉCNICAS DE PRUEBA DE CAJA NEGRA

Existen 5 técnicas y las paso a explicar brevemente para luego mostrar ejemplos:

1. **Clases de equivalencia**

   Se basa en los valores de entrada de una funcionalidad, dividimos los datos en datos de entrada válidos y datos de entrada inválidos, supongamos una aplicación que para utilizarla tienes que ser mayor de edad, con esta técnica tendríamos 2 clases de equivalencia la primera con valores menores de 18 cuyo resultado esperado sería un mensaje que no tienes la edad adecuada para utilizar la aplicación, eso pasa con **Gmail** y con **TikTok** y la otra clase de equivalencia sería de más de 18 años lo que te permitiría acceder al formulario de registro de la aplicación.

   Se tienen que probar las clases al menos una vez (un dato sería 16 años y otro 24 para tener cobertura total en la prueba utilizando clases de equivalencia).

   Se puede utilizar esta técnica en cualquier nivel de prueba que más adelante veremos cuáles son, pero ejemplos son *pruebas unitarias* y *pruebas de integración*.

2. **Análisis de valores limite**

   Se utiliza con los valores límite o frontera de las clases de equivalencia y los valores tienen que poder ser ordenados.

   En el ejemplo anterior hablábamos de una funcionalidad para permitir probar una aplicación, podría ser un desplegable con años de nacimiento o tu fecha de nacimiento, con esta técnica tendríamos que hilar más fino y por ejemplo

podríamos tener otras 2 clases de equivalencia y utilizando esta técnica tendríamos valores alrededor de los valores límite de las clases de equivalencia.

En la tabla de abajo podemos ver un ejemplo de cómo aplicar esta técnica.

	Clase 1	Clase 2	Clase 3	Clase 4
Rangos de valores	<18	>= 18	<0	>110
Valores de datos	17,16,10	18,19,30	-1,0,1	109,110,111
Comportamiento función	No permite acceder a registro	Permite acceder a registro	Valores no permitidos	Valores no permitidos

Esta técnica se puede utilizar en pruebas de cualquier nivel.

3. **Tablas de decisión**

   Es para reglas de negocio complejas y se basa en crear una tabla con todas las posibles entradas y el resultado esperado.

   Se utiliza en todos los niveles de prueba y la cobertura de la prueba se mide por la siguiente fórmula.

   Cobertura prueba= Número de combinaciones que se han probado
   Número de combinaciones totales

4. **Técnicas de pruebas de transición de estados**

   El objetivo es probar los caminos entre estados y que vaya por los estados correctos. Se utilizan diagramas de estados.

5. **Pruebas de caso de uso**

   Se verifica que los requerimientos de los usuarios se cumplen, es muy importante para que luego las pruebas de aceptación no fallen. Es una técnica que lo que hace es probar desde el punto de vista del usuario y prueba los escenarios del negocio para que sean como quieren los usuarios.

1.10.2.2 PRÁCTICAS DE TÉCNICAS DE CAJA NEGRA

1.10.2.2.1 Práctica de clases de equivalencia

Un banco tiene diferentes intereses para cuenta de ahorros dependiendo el dinero que tenga el cliente en la cuenta.

- Para una cuenta que tiene un saldo de 0 a 500 euros el tipo de interés es del 0.5 %.
- Si la cuenta tiene un saldo de más de 500 euros y 9999 euros tiene un tipo de interés del 1.5 %.
- Si la cuenta tiene 10.000 o más euros de saldo el tipo de interés es del 3 %.

- Si la cuenta tiene más de 50.000 euros tiene que estar en una cuenta diferente a la de ahorro, debería ser cuenta hipotecaria para compra de una casa.

Tipo de clases	Clase inválida	Clase válida 1	Clase válida 2	Clase válida 3
Rango de valores	<0 euros	0-500	500-9999	>= 10.000
Valores de datos	-1,-2	200,300	1500,5700	11.000

En esta práctica vemos que hay 4 clases de equivalencia, una clase inválida y otra clase válida; la primera tiene valores inválidos y los tres siguientes valores válidos.

Para que se tenga una cobertura del 100% se tendría que hacer 4 casos de prueba, uno para cada clase de equivalencia.

1.10.2.2.2 Práctica de análisis de valores limite

Si seguimos con la práctica anterior vamos a elegir valores límite alrededor de los valores límite de los rangos y vamos a suponer que si tiene más de 50.000 euros no podrán tener cuenta de ahorro si no otra cuenta distinta y deberá cambiarla, cuenta hipotecaria, por ejemplo.

Tipo de clases	Clase inválida	Clase válida 1	Clase válida 1	Clase válida 2	Clase válida 2	Clase válida 3	Clase inválida
Rango de valores	< 0 euros	0-500	0-500	500-9999	500-9999	>= 10000	>50000
Valores de datos	-1,	0	1	1000	9999	10000	50001

Tendríamos 5 casos de prueba, pero esta técnica se utiliza para elegir los datos alrededor de los límites porque ahí es donde suele haber más errores en el código, utilizamos varios datos para el mismo caso de prueba, pero los casos de prueba serian 5.

1.10.2.2.3 Práctica de tablas de decisión

Se utiliza en un sistema que tiene que reaccionar a una combinación de entradas.

Esta técnica se utiliza sobre todo cuando se tiene que comprobar omisiones en los requerimientos.

Vamos a suponer que tenemos una aplicación de préstamos en que como entrada se recibe la cantidad a pagar mensual y el número de años del préstamo. Las condiciones son el pagaré y la finalización del préstamo.

Al tener dos condiciones donde cada una puede tener los valores de verdadero y falso tenemos 4 combinaciones.

Condiciones	Regla 1	Regla 2	Regla 3	Regla 4
Cantidad para pagar	V	V	F	F
Fin de préstamo	V	F	V	F

Esto significa lo siguiente en la primera regla tenemos como entrada la cantidad a pagar y el fin del préstamo.

En la segunda regla tenemos como entrada solo la cantidad a pagar.

En la tercera regla solo tenemos como entrada el fin del préstamo.

En la cuarta regla no tenemos ninguna entrada.

Visto esto las salidas que tendría el sistema serían las siguientes.

Condiciones	Regla 1	Regla 2	Regla 3	Regla 4
Cantidad para pagar	V	V	F	F
Fin de préstamo	V	F	V	F
Salidas				
Procesar cantidad	V	V		
Procesar fin del préstamo	V		V	

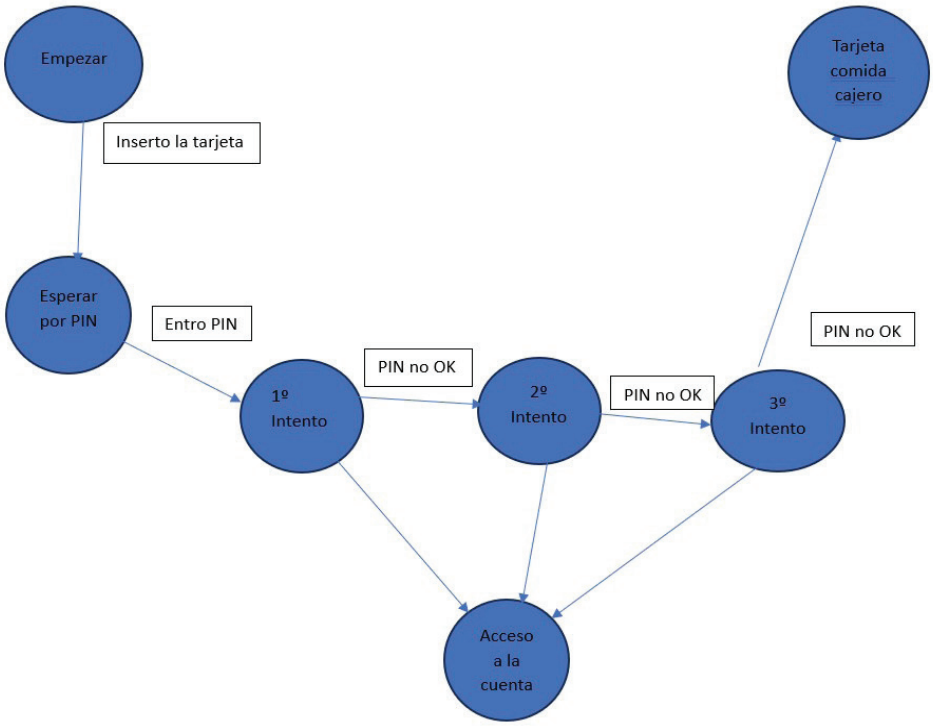
¿Bueno y preguntareis para qué es esto? Pues para saber el número de casos de prueba que necesitamos para probar un sistema como este, en este caso son 4 casos de prueba, uno por cada uno de las reglas o combinaciones que tiene este sistema.

1.10.2.2.4 Práctica de pruebas de transición de estados

Esta técnica se aplica para sistemas que tienen las siguientes características:

- Un número finito de estados.
- Tienen resultados diferentes con las mismas entradas dependiendo del estado anterior.
- Pueden ser representados por diagramas de estados.
- Pueden ser representados con mucho detalle.
- Los modelos de transición de estados están compuestos por los siguientes componentes:
- Los estados en que puede estar la aplicación.
- La transición de un estado a otro.
- Los eventos que causan las transacciones.
- Las acciones que resultan de una transacción.

Un ejemplo de un sistema que se puede representar con un modelo de transición de estados sería retirar dinero de un cajero y su diagrama de estados se puede ver abajo.



1. Acceso a la cuenta al meter el PIN a la primera.
2. Acceso a la cuenta al meter el PIN a la segunda.
3. Acceso a la cuenta al meter el PIN a la tercera.
4. Cajero come la tarjeta.

Con estos casos de prueba recorreríamos todos los estados y tendríamos una cobertura del 100%.

1.10.2.2.5 Práctica de casos de uso

Antes de ver esta técnica debemos entender que es un caso de uso.

1.10.2.2.5.1 Caso de uso

Un caso de uso es la descripción de un uso particular que un usuario también llamado actor le da a un sistema.

El objetivo de los casos de uso es describir todos los usos que se le pueden dar a un sistema al completo, de inicio a fin.

Los actores que interactúan con el sistema pueden ser usuarios, pero también lo pueden ser sistemas y subsistemas.

Los casos de uso se definen en términos de actos, es decir, lo que puede y no puede hacer un usuario en un sistema.

1.10.2.2.5.2 Reglas para la definición de un caso de uso

- Se debe usar un lenguaje de negocio y no técnico.
- Sirven de base para la creación de casos de prueba de aceptación (en la parte de historias de usuario veremos esto).
- Si un caso de uso tiene una precondición debe especificarse.
- Un caso de uso puede tener pos condiciones y deben especificarse.

1.10.2.2.5.3 Ejemplo de caso de uso

Vamos a utilizar el ejemplo de retirar dinero de un cajero que ya vimos en la sección anterior.

Escenario principal **A: Actor** **S: Sistema**	**Pasos**	**Descripción**
	1	A: inserta tarjeta
	2	S: validar tarjeta y pide PIN
	3	A: entra el PIN
	4	S: valida el PIN
	5	S: permite el acceso a la cuenta
Escenario secundario	**2a**	Tarjeta no valida S: rechaza la tarjeta y muestra un mensaje
	4a	PIN no valido Muestra un mensaje y vuelve a pedir el PIN
	4b	PIN erróneo 3 veces Se come la tarjeta y muestra un mensaje

1.10.2.3 TÉCNICAS DE PRUEBA DE CAJA BLANCA

En estas técnicas el objetivo real es el análisis del código fuente y la ejecución de todos los caminos e instrucciones que tiene un código, en general una función que pertenece a una clase.

Una clase es un conjunto de métodos que realizan funcionalidades determinadas, cada método tiene un código que o bien realiza alguna acción o bien devuelve un resultado.

Un ejemplo sería la clase de operaciones aritméticas donde podemos tener los métodos suma resta multiplicación y división, cada uno de esos métodos tiene un código que al ser ejecutado al llamar al método desde otro lugar o desde la misma clase devuelve un valor.

Bien, una vez hablado por encima qué son las técnicas de caja blanca y lo que es una clase y los métodos voy a pasar a describir las técnicas que hay de caja blanca.

Tenemos básicamente 2 técnicas, cobertura de sentencia y cobertura de decisión, que son técnicas para pruebas unitarias.

1.10.2.3.1 Cobertura de decisión

En esta técnica se prueban las sentencias verdaderas y falsas de una condición.

1.10.2.3.2 Cobertura de sentencia

Este concepto es bastante importante porque luego cuando hablemos de pruebas unitarias se hablará mucho de la cobertura de una prueba.

A la cobertura de sentencia también se le llama cobertura de línea y se basa en que se ejecuten las sentencias que sean ejecutables o dicho de otra manera las líneas de código que se puedan ejecutar, por ejemplo, un comentario no se puede, entonces si ejecutas todas las líneas de código de un método la cobertura será del 100%.

Para medir la cobertura de una prueba que realmente sería cobertura de prueba unitaria sería la siguiente formula:

$$Cobertura = \frac{Sentencias\ probadas}{Total\ de\ sentencias}$$

1.10.2.4 PRÁCTICAS DE CAJA BLANCA

1.10.2.4.1 Cobertura de sentencia

Vamos a utilizar un seudocódigo para esta práctica.

```
READ X
READ Y
Z= (X +5) * Y
IF Z > 60 THEN
PRINT Z
END IF
```

En este tipo de técnicas los valores de entrada o valores que le damos a las variables X, Y y Z son muy importantes porque determinan si ejecutamos todas las sentencias o no y el objetivo es una cobertura del 100% que es ejecutar todas las sentencias.

Una diferencia con respecto a cobertura de decisión es que en las condiciones que sería la *IF Z > 60 THEN* solo se contara la parte de condición verdadero, aunque en este caso da igual porque solo hay una sentencia en el caso verdadero no falso.

Prueba 1: X= 5 Y= 4 entonces Z= 40 cobertura del 80% porque son 5 sentencias y se ejecutan 4.

Prueba 2= X= 10 Y = 5 entonces Z= 75 cobertura del 100% porque son 5 sentencias y se ejecutan las 5 al ser TRUE la condición del seudocódigo.

La prueba 2 sería nuestro caso de prueba que nos permitiría con esos datos probar completamente este código que seguramente estaría dentro de un método.

1.10.2.4.2 Práctica de cobertura de decisión

Una característica de la técnica de cobertura de decisión es que cubre tanto las condiciones verdaderas como falsas, en la cobertura de sentencia solo se prueba la condición TRUE por lo que si la condición tiene sentencia tanto para TRUE como para FALSE, aunque nos dé FALSE solo pasaremos por TRUE.

```
IF 5 > 7 THEN
PRINT TRUE
ELSE
PRINT FALSE
```

Con la cobertura de sentencia solo ejecutaríamos *PRINT TRUE* pero con la cobertura de decisión ejecutaríamos las 2.

Por lo tanto, en la cobertura de sentencia el 100% de cobertura implica haber ejecutado todas las sentencias al menos una vez mientras que en la cobertura de decisión implica ejecutar todas las sentencias, pero en las decisiones ejecutar tanto las ramas TRUE como las ramas FALSE incluso si no hay una rama de sentencia FALSE explícita como ocurría en la práctica de cobertura de sentencia.

Ejemplo

```
READ X
READ Y
IF X > Y THEN
Z= 0
END IF
```

En este ejemplo para tener una cobertura del 100% necesitaras.

Casos de prueba con estos datos.

X= 5 Y= 4
X= 4 Y = 5

No hay rama de condición *FALSE,* pero esta técnica ejecuta esa rama *FALSE,* aunque no exista y la tiene en cuenta es decir por la *cobertura de sentencia* tendríamos 4 sentencias por la *cobertura de decisión* tendríamos 5 sentencias.

1.10.2.5 PRUEBAS BASADAS EN EXPERIENCIA

Para poder aplicar esta técnica el conocimiento y experiencia de los **testers** son muy importantes.

Este tipo de pruebas se realizan en aplicaciones de bajo riesgo y cuando los **testers** ya tienen un conocimiento amplio de la aplicación y saben dónde puede fallar y donde tienen que probar, se da muchos en gente que conoce muy bien el negocio como en los bancos y en ministerios públicos como de justicia o sitios así.

Hay básicamente 3 técnicas basadas en la experiencia.

1.10.2.5.1 Predicción de errores

Esta técnica se basa en analizar los errores que ocurrieron en el pasado en esa aplicación o en aplicaciones parecidas para probar si ocurre en la aplicación.

Tener un histórico de los errores que ocurrieron a lo largo del desarrollo y evolución de una aplicación es muy importante porque ayuda a ir a los componentes y funcionalidades donde más errores se dan y ya se tuvo la solución, por lo tanto aumenta la calidad del producto y el tiempo de entrega de un nuevo **release** o versión del producto.

1.10.2.5.2 Pruebas exploratorias

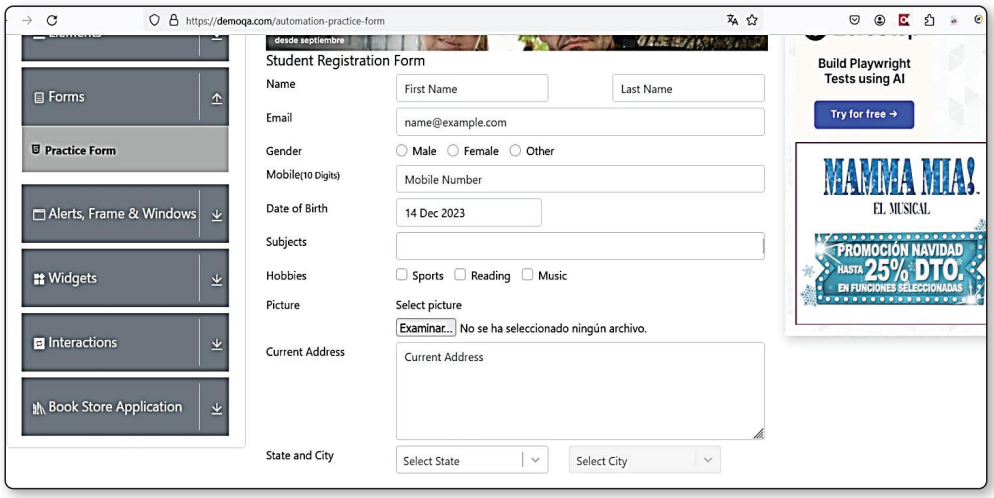
En las pruebas exploratorias no se definen casos de prueba, solo navegamos por la aplicación con el objetivo de conocerla más, no se entrega documentación técnica y de hecho estas pruebas se hacen cuando ocurren tres condiciones.

1. No hay documentación o casi no hay de la aplicación.
2. Se necesita realizar unas pruebas rápidas con el objetivo de ver el estado general de la aplicación.
3. Los **testers** tienen bastante experiencia en las labores QA.

La herramienta que ya vimos **XRay** permite pruebas exploratorias.

1.11 CASO PRÁCTICO. DISEÑO DE UN CASO DE PRUEBA

Vamos a diseñar los casos de prueba del formulario siguiente que está en la página web *https://demoqa.com/automation-practice-form*

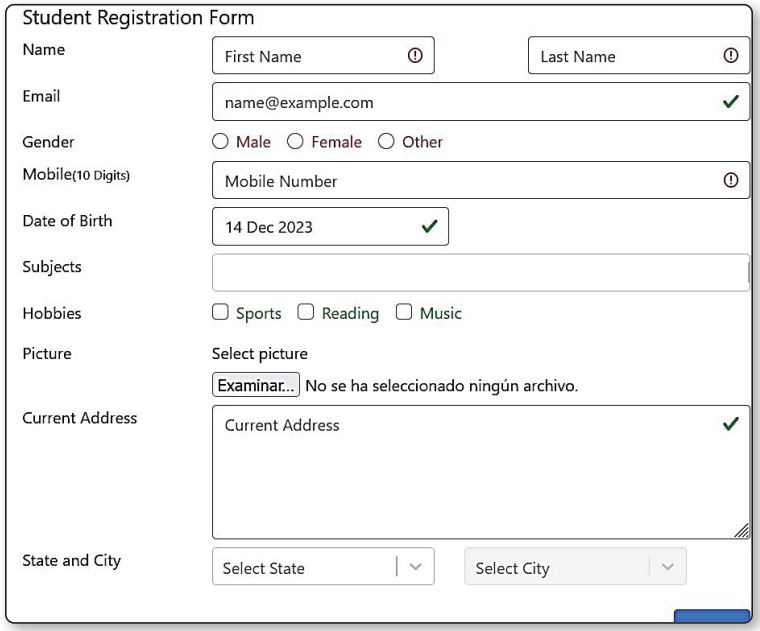


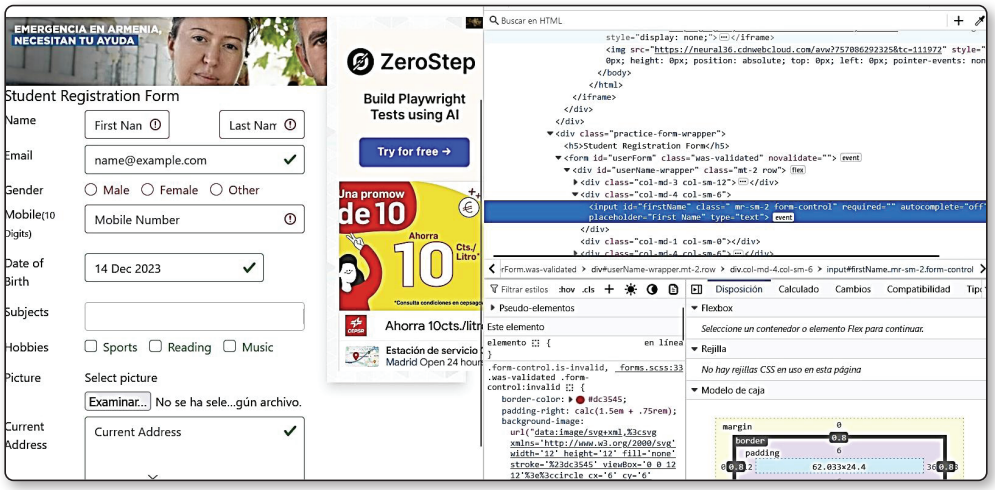
Para eso vamos a utilizar la técnica de las pruebas sintácticas que tiene tres pasos:

1. La creación de las situaciones de prueba basadas en las características de los campos del formulario que pueden ser las siguientes:
   - Tipo de dato, que puede ser numérico, alfanumérico, alfabético.
   - Tamaño del campo.
   - Si el campo tiene valor por defecto.
   - Si el campo es obligatorio.
   - Qué tipo de campo es radiobutton, checkbox, desplegable y si puede seleccionar 1 solo valor o varios valores.
   - Cuáles son los valores máximo y mínimo que puede tener un campo.
   - El campo permite valores especiales como punto coma guion.
   - Si es un campo de fecha qué formato puede tener dd/mm/aa o dd/mm/aaaa o dd-mm-aaaa.
   - Qué atributos tiene el campo en qué posición esta, qué color tiene, qué fuente utiliza, tamaño de la fuente, color.

   Todo esto nos permite crear las situaciones de prueba.
2. Crear los **casos de prueba lógicos** utilizando estas situaciones de prueba.
3. Crear los **casos de prueba físicos**, que sería utilizar los casos de prueba lógicos, pero utilizando valores reales.

Empezamos analizando la página, es página de un registro de estudiante que tiene un montón de campos y los vamos a ir analizando de la siguiente manera, lo primero le damos Enter a un campo, en otros formularios tienes que darle al botón Enviar para que se valide los campos y así saber qué campos son obligatorios porque mostrara un mensaje de error indicando que es necesario, también puedes pulsar botón derecho y darle a la opción de inspeccionar lo que te permitirá ver sus atributos en el lenguaje HTML viendo si es obligatorio o no por el atributo **required.**





Vamos a ir aplicando todos los análisis que vimos de la técnica de prueba sintáctica en este formulario con cada campo, si es obligatorio, tipo de campo, valores máximos y mínimos del campo, si tiene valor por defecto, si permite caracteres especiales, el tipo de campo que es para saber si permite un solo valor o varios y el formato de un campo por si es de una fecha, en este caso no es necesario pero atributos como su posición, su color o la fuente que utiliza podría ser una situación de prueba si es una prueba de diseño pero aquí estamos haciendo una prueba no funcional.

- El campo **First Name** es obligatorio, es un campo de tipo *input* que acepta valores alfanuméricos, permite números letras y caracteres especiales y también espacios en blanco, no tiene un valor por defecto su mínimo tamaño permitido es 1 y parece que no tiene tamaño máximo.
- El campo **Last Name** es obligatorio, es un campo de tipo *input* alfanumérico, permite números letras y caracteres especiales y también espacios en blanco, no tiene un valor por defecto su mínimo tamaño permitido es 1 y parece que no tiene tamaño máximo.
- El campo **Email** no es obligatorio, es un campo de tipo *input*, acepta valores alfanuméricos, no permite caracteres especiales, debe tener un correo electrónico correcto como *canosa1@gmail.com* es decir con usuario, dominio principal y secundario, no tiene valor por defecto, su tamaño mínimo es 7 y parece que no tiene tamaño máximo.
- El campo **Gender** es un campo obligatorio, no tiene valor por defecto, es un campo de tipo *RadioButton*, puede tener un solo valor a la vez y los valores que puede tener son *Male*, *Female* y *Other.*
- El campo **Mobile** es obligatorio, no tiene valor defecto, es un campo de tipo *input*, acepta solo valores numéricos, tiene 10 números como máximo, y diez números como mínimo.
- El campo **Date of Birth** no es obligatorio, tiene el valor por defecto del día que se visita la página, es un campo de tipo *calendar y* su formato es dd mmm yyyy con los meses en inglés, si borras el campo desaparece el formulario esto puede ser un error.
- El campo **Subject** es un campo no obligatorio, de tipo *input* que acepta valores alfanuméricos*, que no* se sabe su tamaño máximo pero su tamaño mínimo es cero porque es optativo.
- El campo **Hobbies** es un campo de tipo *checkbox,* que no es obligatorio, que permite seleccionar varios valores va la vez y que tiene los valores Sports, Reading, Music.
- El campo **Picture** es un campo no obligatorio, de tipo *file y que acepta solo archivos de imágenes.*
- *El campo* **Current Address** es un campo no obligatorio, de tipo *Área, que* acepta valores alfanuméricos y parece que acepta 5 filas de texto.

Situaciones de prueba serian:

1. Registro correcto de todo el formulario.
2. Todos los campos vacíos del formulario.
3. Todos los campos rellenos excepto los campos opcionales.
4. Campo FirstName vacío y los demás rellenos.
5. Campo LastName vacío y los demás rellenos.
6. Campo Email con caracteres especiales.
7. Campo Email sin @
8. Campo Email con un correo de menos de 7 caracteres.
9. Todos los campos rellenos y el campo Gender con valor Male.
10. Todos los campos rellenos y el campo Gender con valor Female.
11. Todos los campos rellenos y el campo Gender con valor Others.
12. Todos los campos rellenos y el campo Gender sin valores marcados.
13. Todos los campos rellenos y el campo Mobile vacío.
14. Todos los campos rellenos y el campo Mobile con texto.
15. Todos los campos rellenos y el campo Mobile con 9 números.
16. Todos los campos rellenos y el campo Subject vacío.
17. .Todos los campos rellenos y el campo Hobbies sin valores marcados.
18. Todos los campos rellenos y el campo Hobbies con los valores Sports y Music marcados.
19. Todos los campos rellenos y el campo Picture vacío.
20. Todos los campos rellenos y el campo Picture con un archivo de tipo Word.
21. Todos los campos rellenos y el campo Current Address vacío.
22. Todos los campos rellenos y el campo Current y el campo Current Address con 6 filas.
23. Borramos el campo Date of Birth.

De todas estas 23 situaciones de prueba las más importantes serían las siguientes.

En la tabla de abajo se ven los casos de prueba lógicas.

Casos de prueba	Titulo	Resultado esperado
CP 1	Registro correcto de todo el formulario	Éxito
CP 2	Todos los campos rellenos excepto los campos opcionales	Éxito
CP 3	Todos los campos vacíos del formulario	Error
CP 4	Campo Email con caracteres especiales	Error
CP 5	Campo Email sin @	Error
CP 6	Campo Email con un correo de menos de 7 caracteres	Error
CP 7	Todos los campos rellenos y el campo Gender sin valores marcados	Error
CP 8	Todos los campos rellenos y el campo Mobile vacío	Error
CP 9	Todos los campos rellenos y el campo Mobile con texto	Error
CP 10	Todos los campos rellenos y el campo Mobile con 9 números	Error
CP 11	Todos los campos rellenos y el campo Picture con un archivo de tipo Word	Éxito
CP 12	Borramos el campo Date of Birth	Error

Y abajo tendremos la tabla con los 11 casos de prueba físicos que serían con el identificador, título de caso de prueba, pasos, los valores y el resultado esperado.

Caso de prueba	Titulo	Pasos	Valores	Resultado
CP 1	Registro correcto de todo el formulario	Rellenar todos los campos con valores correctos y pulsar la tecla Enter	Name: Manuel Ferreiro Email: Manuel.ferreiro@gmail.com Gender: male Mobile: 6674567450 Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: music Picture: C:/imagen.jpg Current Address: esto es una prueba	El formulario no devuelve ningún mensaje de error

CP 2	Todos los campos rellenos excepto los campos opcionales	Rellenar todos los campos obligatorios con valores correctos y pulsar la tecla Enter	Name: Manuel Ferreiro Email: Manuel.ferreiro@gmail.com Gender: male Mobile: 6674567450	El formulario no devuelve ningún mensaje de error
CP 3	Todos los campos vacíos del formulario	Dejar todos los campos vacíos y pulsar Enter		Los campos obligatorios se ponen en rojo y en los campos de tipo input aparece un círculo con una exclamación en rojo
CP 4	Campo Email con caracteres especiales	Rellenar todos los campos con valores correctos y el campo Email con caracteres especiales y pulsar la tecla Enter	Name: Manuel Ferreiro Email: Manuel<$ferreiro@gmail.com Gender: male Mobile: 6674567450 Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: Music Picture: C:/imagen.jpg Current Address: esto es una prueba	El campo Email se pone en rojo y con un círculo rojo con signo de exclamación
CP 5	Campo Email sin @	Rellenar todos los campos con valores correctos y el campo Email sin arroba y pulsar la tecla Enter	Name: Manuel Ferreiro Email: Manuel.ferreirogmail.com Gender: male Mobile: 6674567450 Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: music Picture: C:/imagen.jpg Current Address: esto es una prueba	El campo Email se pone en rojo y con un círculo rojo con signo de exclamación

CP 6	Campo Email con un correo de menos de 7 caracteres	Rellenar todos los campos con valores correctos y el campo Email con un correo de 7 caracteres y pulsar la tecla Enter	Name: Manuel Ferreiro Email:a@msn.c Gender: male Mobile: 6674567450 Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: music Picture: C:/imagen.jpg Current Address: esto es una prueba	El campo Email se pone en rojo y con un círculo rojo con signo de exclamación
CP 7	Todos los campos rellenos y el campo Gender sin valores marcados	Rellenar todos los campos con valores correctos y el campo Gender sin valores marcados y pulsar la tecla Enter	Name: Manuel Ferreiro Email: a@msn.c Gender: sin marcar Mobile: 6674567450 Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: music Picture: C:/imagen.jpg Current Address: esto es una prueba	Los radiobutton Male Female y Others se ponen en color rojo
CP 8	Todos los campos rellenos y el campo Mobile vacío	Rellenar todos los campos con valores correctos y el campo Mobile vacío y pulsar la tecla Enter	Name: Manuel Ferreiro Email: Manuel.ferreiro@gmail.com Gender: Male Mobile: Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: Music Picture: C:/imagen.jpg Current Address: esto es una prueba	El campo Mobile se pone en rojo y con un círculo rojo con signo de exclamación
CP 9	Todos los campos rellenos y el campo Mobile con texto	Rellenar todos los campos con valores correctos y el campo Mobile con texto y pulsar la tecla Enter	Name: Manuel Ferreiro Email: Manuel.ferreiro@gmail.com Gender: Male Mobile: tres cuatro seis Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: Music Picture: C:/imagen.jpg Current Address: esto es una prueba	El campo Mobile se pone en rojo y con un círculo rojo con signo de exclamación

CP 10	Todos los campos rellenos y el campo Mobile con 9 números	Rellenar todos los campos con valores correctos y el campo Mobile vacío y pulsar la tecla Enter	Name: Manuel Ferreiro Email: Manuel.ferreiro@gmail.com Gender: Male Mobile: 6674567450 Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: Music Picture: C:/imagen.jpg Current Address: esto es una prueba	El campo Mobile se pone en rojo y con un círculo rojo con signo de exclamación
CP 11	Todos los campos rellenos y el campo Picture con un archivo de tipo Word	Rellenar todos los campos con valores correctos y el campo Picture se selecciona un archivo Word	Name: Manuel Ferreiro Email: Manuel.ferreiro@gmail.com Gender: Male Mobile: 6674567450 Date of Birth: 14 Nov 1977 Subjects: prueba Hobbies: Music Picture: C:/archivo.doc Current Address: esto es una prueba	El formulario muestra un mensaje de archivo no permitido
CP 12	Borramos el campo Date of Birth	Rellenar todos los campos con valores correctos y borramos el campo Date of Birth	Name: Manuel Ferreiro Email: Manuel.ferreiro@gmail.com Gender: Male Mobile: 6674567450 Date of Birth: Subjects: prueba Hobbies: Music Picture: C:/imagen.jpg Current Address: esto es una prueba	El campo de Date of Birth se borra

Se puede ver como el diseño de casos de prueba físico y lógico, es que en el primero utilizamos valores reales, indicamos los pasos y el resultado esperado y en el otro no, solo indicamos el nombre y si es un caso de prueba que valida un caso de éxito o un caso de no éxito que debe ser controlado con algún tipo de mensaje.

En los casos de prueba 11 y 12 debería ocurrir el resultado esperado, pero no ocurre y es cuando es un defecto del software que tendríamos que registrar en una herramienta de gestión de defectos como **Mantis o Jira** y hacer un seguimiento hasta que se resuelve el defecto haciendo un ciclo de pruebas.