«Talento Tech»

Testing QA

Clase 02

Buenos Aires Ciudad

Clase N°2 | Proceso de QA

Temario:

- Metodologías Tradicionales y Ágiles
    - Scrum
    - Fundamentos de Scrum (roles, artefactos, ceremonias)
    - El rol del Tester dentro de un equipo Scrum
- Principios del testing
- ¿Cuáles son las responsabilidades de un Tester Analista de Calidad?
- Objetivos de Probar
    - Planificación de pruebas dentro de los sprints
- Validar vs. Verificar
- Requerimientos como Documentación de entrada
    - Requerimientos
    - Épicas
    - Features
    - User Stories
- Definition of Done y criterios de calidad
- Presentación del Proyecto a desarrollar durante el curso

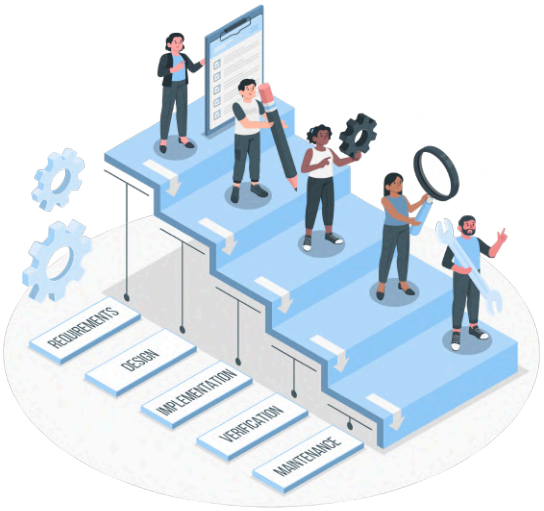
Objetivos de clase

En esta clase, conectaremos los puntos clave del Control de Calidad del Software. Comenzaremos explorando las metodologías de desarrollo y cómo influyen en el testing. Luego, nos sumergiremos en los principios del testing, las responsabilidades de un Tester Analista de Calidad y los objetivos de las pruebas. Aprenderemos a diferenciar entre validar y verificar requerimientos, y cómo los requerimientos (épicas, features, user stories) se utilizan como guía para el testing. Al finalizar la clase, comprenderán cómo todos estos conceptos se entrelazan para garantizar la calidad del software.

Metodologías de desarrollo: tradicional y ágil

En el mundo del desarrollo de software, existen diferentes maneras de abordar un proyecto, desde la planificación inicial detallada hasta la adaptación constante a los cambios. Estas distintas formas de trabajar se conocen como enfoques de desarrollo. En esta clase, exploraremos dos de los enfoques más utilizados: el enfoque tradicional (con ejemplos como Waterfall) y el enfoque ágil (con ejemplos como Scrum o Kanban), utilizando analogías culinarias para comprender sus características, ventajas y desventajas de manera sencilla y amena.

Metodología Tradicional (Waterfall)



Imaginate un proyecto como hacer una torta siguiendo la receta de un libro: ¡hay que hacer cada paso en orden y no te podés saltear ninguno! Primero integras los ingredientes secos, después los húmedos, horneás y al final decorás. Los ingredientes (requisitos) ya están definidos y no podés andar cambiando nada a mitad de camino.

- **Lo bueno**:
    - **Es claro**: Sabés bien qué tenés que hacer en cada momento.
    - **Tenés el control**: Podés seguir el paso a paso sin problemas.
    - **Queda todo anotado**: ¡Como un recetario!
- **Lo malo**:
    - **Es rígido**: ¡Ay si te equivocás en un paso o querés cambiar un ingrediente!
    - **Es lento**: No probás la torta hasta el final.
    - **Es riesgoso**: Si algo sale mal, ¡hay que empezar de nuevo!

Metodología Agile (Scrum, Kanban, etc.)

Las metodologías de desarrollo de software definen cómo se gestiona el proceso de construcción de una aplicación o sistema. Existen enfoques tradicionales y ágiles, cada uno con sus características y ventajas.

Ejemplo: el proyecto es como cocinar con un amigo al que le gusta improvisar: ¡van probando y ajustando la receta sobre la marcha! Primero hacen la masa, después prueban el relleno, ven si les gusta y le agregan más condimentos. El cliente (el que va a comer la torta) participa y prueba cada etapa.

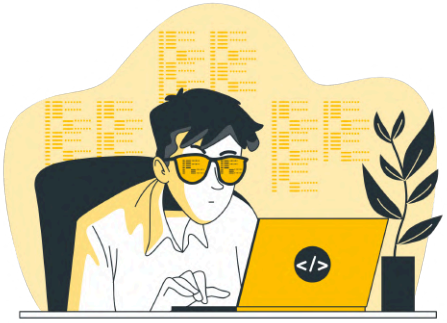
- **Lo bueno**:
    - **Es flexible**: ¡Se adapta a los cambios como un guante! Si quieren ponerle dulce de leche en vez de crema, ¡no hay problema!
    - **Es rápido**: Prueban la torta en cada etapa, así se aseguran de que esté rica.
    - **Es colaborativo**: ¡Es como cocinar juntos y divertirse!
- **Lo malo**:
    - **Es incierto**: Al principio, no saben bien cómo va a quedar la torta final.
    - **Depende del equipo**: ¡Los dos tienen que ser buenos cocineros para que salga bien!
    - **A veces se olvidan de anotar la receta**: ¡Priorizan que la torta esté rica y se olvidan de los detalles!

¿Cuál elegir?

- **Tradicional**: Si tenés los ingredientes claros y querés seguir la receta al pie de la letra.
- **Agile**: Si te gusta probar cosas nuevas y querés que la torta se adapte a tus gustos.

La metodología tradicional es como hacer una torta siguiendo un libro de recetas, mientras que Agile es como cocinar con un amigo al que le gusta innovar y probar cosas nuevas.

Scrum



Scrum es un marco de trabajo ágil que permite entregar software de alta calidad de manera iterativa e incremental. Se basa en la colaboración, la adaptabilidad y la mejora continua.

Fundamentos de Scrum:

Scrum se estructura en torno a tres pilares fundamentales:

- **Transparencia:** Todos los involucrados tienen visibilidad del proceso y el trabajo en curso.
- **Inspección:** Se revisan periódicamente los avances para detectar posibles mejoras.
- **Adaptación:** Se ajusta el trabajo en función de lo aprendido en cada iteración.

Roles en Scrum

1. **Product Owner:** Representa la voz del cliente y prioriza el trabajo según el valor que aporta al negocio.
2. **Scrum Master:** Facilita el proceso Scrum y ayuda al equipo a eliminar obstáculos.
3. **Development Team:** Equipo multidisciplinario encargado de desarrollar, probar y entregar el producto.

Artefactos en Scrum:

- **Product Backlog:** Lista priorizada de funcionalidades y requerimientos.
- **Sprint Backlog:** Conjunto de tareas seleccionadas para un Sprint.
- **Incremento:** Producto funcional entregable al final de cada Sprint.

Ceremonias en Scrum:

1. **Sprint Planning:** Planificación de las tareas a desarrollar en el Sprint.
2. **Daily Scrum:** Reunión diaria de sincronización del equipo.
3. **Sprint Review:** Demostración del trabajo completado.
4. **Sprint Retrospective:** Análisis de mejoras en el proceso.

El rol del Tester dentro de un equipo Scrum

El tester en un equipo Scrum no se limita a ejecutar pruebas al final del desarrollo, sino que participa en todo el ciclo de vida del Sprint. Sus principales responsabilidades incluyen:

- Revisar y comprender las User Stories y sus criterios de aceptación.
- Colaborar con el Product Owner y el equipo para definir requerimientos testables.
- Diseñar y ejecutar casos de prueba de manera continua dentro del Sprint.
- Automatizar pruebas para acelerar la retroalimentación.
- Validar la calidad del software en cada incremento, asegurando que cumple con los estándares definidos.

Principios del Testing

Estos son algunos principios básicos que guían el trabajo de los testers:

- **Las pruebas muestran la presencia de defectos:** Las pruebas nos ayudan a encontrar errores, pero no pueden garantizar que no haya ninguno. Es como buscar agujas en un pajar: podemos encontrar algunas, pero no podemos estar seguros de haberlas encontrado todas.
- **No es posible realizar pruebas exhaustivas:** No podemos probar todas las combinaciones posibles de un software. Sería como tratar de probar todas las posibles combinaciones de contraseñas para una cuenta de correo electrónico: ¡imposible!
- **Pruebas tempranas:** Cuanto antes empecemos a probar, más fácil y barato será corregir los errores. Es como revisar los planos de una casa antes de empezar a construir: si encontramos un error en los planos, será mucho más fácil y barato corregirlo que si lo encontramos una vez que la casa ya está construida.
- **Agrupamiento de defectos:** Los errores suelen agruparse en zonas específicas del software. Es como cuando encontramos un nido de hormigas: es probable que haya más hormigas cerca.
- **Paradoja de los pesticidas:** Si siempre usamos las mismas pruebas, llegará un momento en que ya no encontraremos errores. Hay que variar las pruebas, como cuando cambiamos de cebo para pescar.
- **Las pruebas dependen del contexto:** No es lo mismo probar una app de juegos que un software para un banco. Cada software tiene sus propias características y riesgos.
- **Falacia de ausencia de errores:** Que no hayamos encontrado errores no significa que el software sea perfecto. Puede haber errores que aún no hemos descubierto.

¿Cuáles son las responsabilidades de un Tester Analista de Calidad?

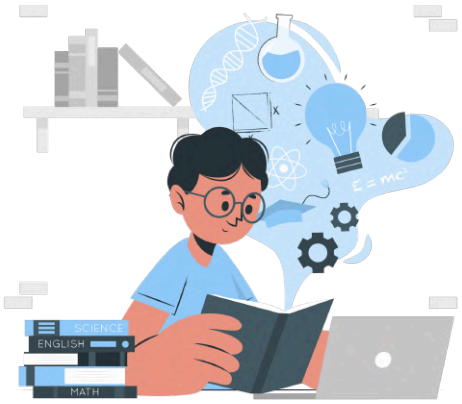
Los Testers Analistas de Calidad son los encargados de asegurar que el software sea de calidad. Trabajan codo a codo con los desarrolladores, analistas de negocio y otros involucrados para entender los requisitos y garantizar que el producto final cumpla con lo que se espera.

Responsabilidades clave:

- **Análisis de requisitos:** Estudiar y comprender qué necesita el cliente y qué debe hacer el software.
- **Planificación y diseño de pruebas:** Crear casos de prueba que cubran diferentes escenarios y funcionalidades del software. Un caso de prueba es como una receta: describe los pasos a seguir para probar una funcionalidad específica.
- **Ejecución de pruebas:** Poner a prueba el software siguiendo los casos de prueba diseñados y registrar los resultados.
- **Informes y gestión de defectos:** Comunicar los errores encontrados a los desarrolladores para que los corrijan. Un defecto es como un error en la receta: algo que impide que el plato salga como se espera.
- **Automatización de pruebas:** Crear pruebas automáticas para ahorrar tiempo y esfuerzo. Es como tener un robot de cocina que siga la receta automáticamente.
- **Pruebas de rendimiento:** Evaluar cómo responde el software bajo diferentes cargas de trabajo. Es como probar si una olla a presión es capaz de cocinar bajo presión sin explotar.
- **Pruebas de seguridad:** Identificar vulnerabilidades en el software que puedan ser aprovechadas por hackers. Es como revisar si una casa tiene cerraduras seguras y ventanas protegidas.
- **Pruebas de usabilidad:** Evaluar si el software es fácil de usar y si la experiencia del usuario es buena. Es como probar si una silla es cómoda y fácil de usar.
- **Pruebas de regresión:** Verificar que los cambios en el software no hayan introducido nuevos errores. Es como volver a probar un plato después de haber modificado la receta para asegurarnos de que sigue saliendo bien.
- **Colaboración y comunicación:** Trabajar en equipo con otros profesionales para lograr un software de calidad. Es como trabajar en equipo en la cocina para preparar una comida deliciosa.

Objetivos de Probar

El objetivo principal de las pruebas de software es entregar un producto confiable, efectivo y de alta calidad.



Objetivos específicos

- **Asegurar la calidad del software:** Esto implica tanto verificar que el software se construyó correctamente (verificación) como asegurar que cumple con las expectativas del cliente (validación). Es como verificar si la casa se construyó según los planos y si cumple con las necesidades del cliente.
- **Identificar y prevenir defectos:** Encontrar la mayor cantidad de errores posibles antes de que el software llegue a los usuarios, y aprender de ellos para evitar cometerlos en el futuro. Es como encontrar todas las imperfecciones en un plato antes de servirlo y aprender de los errores en la receta para mejorarla la próxima vez.
- **Garantizar atributos de calidad:** Asegurar que el software cumpla con los estándares de calidad en cuanto a funcionalidad, rendimiento, usabilidad, seguridad, etc. Es como asegurar que el plato cumpla con los estándares de calidad en cuanto a sabor, presentación, textura, etc.
- **Gestionar riesgos:** Identificar y evaluar los riesgos asociados a posibles fallas del software. Es como evaluar los riesgos de que la olla a presión explote si no se usa correctamente.

Proceso de Validación y Verificación

El proceso de validación y verificación se lleva a cabo antes, durante y después del desarrollo del software para asegurar que cumple con lo especificado y hace lo que se espera.

- **Verificación:** Se enfoca en comprobar que el software cumple con su especificación. La pregunta clave es: ¿Estamos construyendo el producto correctamente? Es como verificar si la casa se construyó según los planos.
- **Validación:** Busca asegurar que el software que se va a construir es el que realmente se necesita y cumple con las expectativas del cliente. La pregunta clave es: ¿Estamos construyendo el producto correcto? Es como verificar si la casa cumple con las necesidades del cliente.

En resumen: La verificación se centra en el proceso de desarrollo, mientras que la validación se centra en el producto final y su adecuación a las necesidades del cliente.

Requerimientos como documentación de entrada

Los **Requerimientos** son las definiciones de alto nivel que los usuarios del sistema definen al principio. Los Analistas de Negocio (BA) trabajan sobre estos requerimientos para detallarlos y hacerlos más específicos.

En proyectos ágiles, los requerimientos se organizan en diferentes niveles:

- **Épicas:** Son los requerimientos más grandes, que abarcan varias funcionalidades. Piensa en ellas como el título de un libro que abarca varios capítulos.
    - **Ejemplo de la vida real:** "Construcción de la casa" (Abarca todo el proyecto de construcción).
    - **Ejemplo de software:** "Gestión de recetas" (Abarca todas las funcionalidades relacionadas con las recetas).

- **Features (Funcionalidades):** Son partes más pequeñas de una épica, que aportan valor por sí mismas. Son como los capítulos de un libro.
    - **Ejemplo de la vida real:** "Construcción de la cocina" (Es una parte importante de la casa que aporta valor por sí misma).
    - **Ejemplo de software:** "Búsqueda de recetas por ingredientes" (Es una funcionalidad específica que aporta valor al usuario).

- **User Stories (Historias de Usuario):** Son la unidad más pequeña de trabajo en Agile. Describen una funcionalidad desde la perspectiva del usuario. Son como las oraciones dentro de un capítulo.
    - **Ejemplo de la vida real:** "Como miembro de la familia, quiero una cocina con una isla central para poder cocinar y compartir con mi familia" (Describe una necesidad específica del usuario).
    - **Ejemplo de software:** "Como usuario, quiero poder buscar recetas ingresando los ingredientes que tengo en casa para encontrar opciones de comida" (Describe una necesidad específica del usuario).

Ejemplo de documento de requerimientos

Veamos un ejemplo más concreto:

Imaginá que se está desarrollando una aplicación de reservas de hotel. Nuestra historia se inicia definiendo grandes capítulos (épicas) que reflejan las metas principales del usuario. En este caso, tenemos dos grandes capítulos: Reserva de habitaciones y Check-in en línea.

Capítulo 1: Reserva de Habitaciones
El usuario inicia su viaje seleccionando una habitación, llenando un formulario, recibiendo una confirmación y realizando un pago seguro. Cada paso se documenta como una user story para garantizar que la experiencia sea fluida y confiable.

Capítulo 2: Check-in en Línea
Una vez reservada la habitación, el huésped ingresa sus datos, recibe una confirmación, obtiene un código QR para acceder a su habitación y el sistema valida toda la información.

Al plasmar estos pasos en un Excel con épicas, features y user stories, el equipo de QA tiene un mapa claro para probar cada detalle, asegurando una experiencia sin contratiempos desde la reserva hasta el check-in.

Cada uno de estos pasos no solo define lo que el usuario espera, sino que también establece los puntos de verificación que el equipo de QA debe probar. De esta forma, se asegura que cada interacción funcione según lo planeado.

Planificación de pruebas dentro de los sprints

La planificación de pruebas en metodologías ágiles se realiza de manera iterativa dentro de cada Sprint. A diferencia de los enfoques tradicionales, en Scrum las pruebas se integran en todo el proceso de desarrollo.

En Scrum, un Sprint es un período de tiempo fijo (generalmente de 1 a 4 semanas) en el que el equipo trabaja para completar un conjunto de tareas previamente planificadas. Durante un Sprint, el equipo desarrolla, prueba y entrega un incremento funcional del producto. Cada Sprint sigue un ciclo estructurado con etapas bien definidas:

Fases de la planificación de pruebas en un Sprint:

1. **Sprint Planning:**
    - Se identifican los criterios de aceptación de cada User Story.
    - Se definen los tipos de pruebas a ejecutar: funcionales, regresión, automatización, etc.
2. **Desarrollo y ejecución de pruebas:**
    - Se diseñan y ejecutan pruebas de manera continua junto con el desarrollo.
    - Se identifican defectos y se reportan para su corrección en el mismo Sprint.
3. **Sprint Review y Retrospective:**
    - Se validan los resultados de las pruebas y se analiza la calidad del producto.
    - Se identifican oportunidades de mejora en el proceso de pruebas.

Presentación del Proyecto a desarrollar durante el curso

¡Trabajando en Talento Lab!

Tu primera responsabilidad en Talento Lab consistirá en:

- Informarte de la metodología a utilizar en el proyecto de Talento Lab.
- Las herramientas a utilizar
- Entender el objetivo del proyecto
    - Informandote y revisando los requerimientos
    - Verificando que cumplan con todas las condiciones
    - Pedir que se completen los requerimientos para garantizar que la cobertura de las pruebas es la correcta.

En esta etapa, vos tenés que asegurarte de que los requisitos y las historias de usuario tengan todo lo necesario para que podamos armar las pruebas. Estas pruebas nos van a decir si se cumplen los Criterios de Aceptación de cada User Story. Y también tenés que ver si con esas definiciones alcanza para que el cliente o usuario reciba justo lo que pidió.

¿Cómo trabajarás en Talento Lab?

Como **Analista de Calidad Trainee** recién contratado en **Talento Lab**, te citaran a reuniones con el Product Owner y el resto del equipo de QA (Matias) así como los desarrolladores y el Scrum Master.

En estas reuniones, Silvia explicará la metodología ágil utilizada en el proyecto y, junto con Matías, revisarán un requerimiento específico para demostrar cómo se desglosa en User Stories (pequeñas porciones de funcionalidad desde la perspectiva del usuario).

Luego de esta revisión inicial, se te asignará un conjunto de requerimientos para que los examines y determines si su desglose en User Stories es adecuado. Deberás verificar que los criterios de aceptación de cada User Story garanticen una cobertura completa de las pruebas necesarias.

Ejercicio Práctico

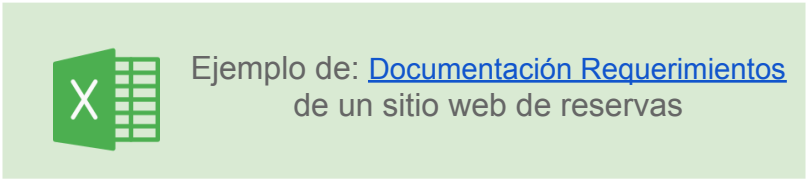
Ejercicio: Desgranamiento de los requerimientos con técnicas básicas

Principios – Documentación de Entrada

Sitio Web Talento Lab:

https://talentolab-test.netlify.app/

- **Instrucciones:**
    1. Selecciona dos funcionalidades sencillas del proyecto (ej.: Que todos los enlaces nos llevan a las distintas secciones, cargar CV en la página web).
    2. Diseña los Requerimientos en formato Agile:
        - Épicas (Requerimiento Global)
        - Features (Funcionalidades)
        - User Stories (Historias de Usuario)
    3. Documenta las Épicas, Features y User Stories manualmente en una hoja de cálculo. (Puedes usar la hoja de ejemplo)
- **Objetivo:** Aplicar técnicas fundamentales en el desgranamiento de requerimientos



Preguntas para Reflexionar:

- ¿Considerás que la calidad del software es importante para una empresa? ¿Por qué?
- ¿Crees que las empresas valoran el trabajo de los Analistas de Calidad? ¿Qué beneficios aporta el QA a una empresa?

Próximos Pasos

En la próxima clase, vamos a ver cómo se definen los criterios de aceptación para las User Stories y cuál es el ciclo de vida de QA


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