Componente Teórico (70 minutos)

1. Desarrollo de Software: Metodologías Tradicionales vs Ágiles

R/ Son dos enfoques que se diferencian en la estrategia de abordar un proyecto, hablando específicamente de un proyecto de software.

En las tradicionales se abarca desde una perspectiva etapa por etapa, documentación, diseño, implementación, pruebas y entrega y proyectado en medianos y largos periodos de trabajo, por cual se cuentan con metodologías muy robustas para mitigar los riesgos entre estas etapas, ya que el ajustar un requerimiento en una etapa de implementación es muy costoso para el proyecto, en tiempo y recursos.

Las metodologías agiles tienen un enfoque en no pensar en una sola entrega sino en múltiples entregas, se trabajan sobre iteraciones en donde las etapas no tardan tanto tiempo entre cada una, y se le permite entregar valor más temprano al cliente y por lo tanto al usuario

Definición personal de cómo es el proceso de desarrollo de software utilizando metodologías tradicionales y ágiles.

2. Rol del Analista de Calidad de Software en Metodologías Ágiles

R/ Como analista de calidad he sentido que las metodologías ágiles aplicadas de forma aplicada y disciplinada, funcionan mejor en cuanto:

- Al time to market: de acuerdo a que permite una adaptación pronta de los requerimientos al software, y esto les da una ventaja competitiva a los clientes.
- Trabajo en equipo: al permitir más sinergias y siendo la comunicación un pilar muy importante, se permite identificar mayores oportunidades de mejora tanto al nivel colectivo como individual
- Entrega de valor temprana: esto le permite al cliente un retorno de valor más pronto de su inversión.

En cuanto a las tradicionales tienen de ventaja el estudio metódico de lo requerimiento y posiblemente identificar mayores oportunidades estratégicas en cuanto a que se cuenta con mayor tiempo para el análisis.

Descripción del rol de analista de calidad de software en proyectos ágiles.

3. Diferencias Clave

1. Aplicar calidad temprana: al entrar desde la etapa de definición, permite identificar de manera más oportuna errores que se pueden cometer desde el análisis de la situación problémica que da paso al planteamiento de la solución con cada requerimiento.
2. Automatización de pruebas: en cada una de las entregas que se van realizando el analista de pruebas tiene mayor oportunidad de identificar oportunidades de automatización, en los diferentes niveles de prueba del desarrollo
3. Aportar más al trabajo en equipo: en temas de comunicación y el día a día las metodologías agiles le brindan mayor oportunidad de aportar en las dinámicas los equipos.

A. Verificar vs Validar

R/ Verificar es que cumple con una serie de criterios definidos. Mientras que validar es que dado un procedimiento de un contexto, como ejemplo esta lo legal o contable, se cumple con estas generalidades.

B. Issue vs Bug

R/Un issue se refiere ampliamente a un artefacto con una característica especial puede ser una historia, un mismo bug, o una épica. El mismo se refiere a un error encontrado.

C. Novedad vs Hallazgo

R/ Una novedad está relacionado con algún evento, información o situación con el cual no se contaba en el equipo. El hallazgo está relacionado con información encontrado luego de un análisis de la ejecución de un proceso determinado, los hallazgos son los resultados de esos análisis que se tenían como objetivo

4. Explique los entregables de las Etapas de Desarrollo

A. Planeación:

R/ Plan de pruebas: contiene la generalidad, la estrategia, y los pasos necesarios y requeridos al realizar el proceso de pruebas, se contemplan, herramientas, ambientes de desarrollo, matriz de prueba, y también la forma como se entregarán los resultados para dar por contemplado la finalización del proceso de pruebas.

B. Diseño de Escenarios y Casos de Prueba:

R/ Son cada uno de las especificaciones con el paso a paso con su respectivo resultado esperado, que contiene las verificaciones que se realizan para dar por contemplado y finalizado un proceso de pruebas.

Esto se almacena en una suite de pruebas, ordenados por su respectivas historias épicas e historias de usuario

C. Ejecución:

R/ Acá ya sea de forma manual o automatizada, se realiza la ejecución de los escenarios diseñados.

Esta debe contar con su respectivo respaldo en evidencias ordenadas.

D. Reporte de Hallazgos:

R/ De acuerdo a la ejecución se organiza un reporte en donde están los casos exitosos, fallidos y los respectivos incidentes encontrados.

E. Feedback:

R/ Se da una retroalimentación de los resultados de las pruebas con sus respectivas oportunidades de mejoras sobre el equipo como la ejecución del equipo de QA.

5. Herramientas de Gestión de Pruebas

Experiencia con herramientas específicas y preferencias personales.

R/ Yo he trabajado con Xray y zeplin de jira, y team foundation server, prefiero trabajar con zeplin por la forma en que quedan ordenados los escenarios de prueba, me parece que generar mas orden y flexibilidad, y tuve la oportunidad de trabajar con la generación de informes a partir de las ejecuciones permitiendo automatizar eficientemente el proceso de pruebas.

6. Eventos Scrum

Descripción de actividades en cada evento Scrum.

R/ Inception: obtención de las épicas a atacar en el proyecto.

Refinamiento: determinar los criterios de aceptación y organizar las historias a trabaja.

Planing: Comprometer las historias a abordar durante el sprint.

Review: Presentar los resultados de los desarrollado completados en el sprint

Retrospectiva: Socializar la experiencia y las dinámicas de equipo dentro del sprint con la idea de mejoras, y presentar oportunidades de mejora para el equipo.

7. Automatización y Programación Orientada a Objetos (POO)

A. Pilares de POO en Automatización: Listado y explicación.

R/

- Abstracción: Proceso en el cual se permita ordenar funciones y características de un objeto en una clase.
- Herencia: Es un mecanismo que posee la programación orientada a objetos, para permitir extender comportamientos y atributos de una clase que posee varias características similares.
- Encapsulamiento: Es la propiedad que tienen los objetos de guardarse la privacidad de ciertos atributos y métodos dentro de un área determinada (entre clases del mismo paquete, solo para sí mismo).
- Polimorfismo: Significa la habilidad que tiene un objeto para realizar un método de diferentes formas, dependiendo del tipo de dato o clase que sea llamado; más específicamente definido como la habilidad de redefinir un método para clases derivadas.

B. Principios SOLID: Su papel en la automatización de pruebas.

R/

Son una serie de principios establecidos en programación orientada a objetos, para eliminar alto acoplamiento y ofrecer mayor extensión y mantenibilidad al código

S(single responsability): cada clase le correponde su resposabilidad

O(open/closed principle): abierto en extensión, cerrado en modificación de clases

L(Principio de sustitución Liskov): consiste que si en un caso de se requiere reemplazar un hijo por el padre, debe de seguir funcionando y cada resposabilidad la debe de asumir la clase

I(interface segregation principle): No se debe de adicionar a una interface existente nuevos métodos, en caso de que exista la necesidad de más funcionalidades, lo que debería es agregar otra interface que contenga esa funcionalidad, y se debe de separar resposabilidades por interface

D(Dependency inversion principle): es una manera de desacoplar módulos y en donde los detalles deben depender de la abstracción, y no la abstracción de los detalles; también hace referencia directa a que los módulos, no deben de ser influidos por las clases y subclases.Lo esencial es que si la clase A que tiene métodos que funcionan para la clase B, al realizar cambios sobre la clase A, la clase B no tenga que enterarse y pueda seguir trabajando con normalidad (baja acoplabilidad)

Su importancia para la automatización de pruebas es que permite tener proyectos, con un ordenamiento, que permiten que sean consistentes en el tiempo, y que el mantenimiento de estos no signifique un proceso que de alguna manera vaya desperdiciarse el trabajo realizado en su construcción.

C. Escenarios Críticos: Criterios para su definición.

R/ Esta determinación se debe de realizar con los flujos esenciales para el negocio, y estos se deben de construir desde una visión que su ejecución no es negociable ante una salida a producción con nuevos features.

D. Pirámide de Automatización de Cohn: Definición y explicación.

R/Es una pirámide que consta con cada uno de los niveles de pruebas que se prueban en un proceso de pruebas.

Está construida con el ánimo de ilustrar los esfuerzos y la priorización que se deben realizar en este de acuerdo a los niveles de prueba:

- Unitarias: están pruebas son las mas relevantes y es donde mas esfuerzo se debería colocar, y debe estar totalmente automatizado.
- Integración: son las pruebas que se realizan para revisión de los componentes del aplicativo, estás también deben de ser automatizadas.
- Automatización de pruebas E2E: estas son las que se construyen de acuerdo al flujo crítico del aplicativo, estas solo deben enfocarse en la alta criticidad, debido a que el esfuerzo en mantenimiento es alto, al ser las interfaces gráficas más vulnerables a sufrir cambios
- Pruebas manuales: estas son las de menos prioridad, dado los altos riesgos que conllevan dados el gran esfuerzo y el conocimiento humano inherente.

8. Servicios

A. Definición de Servicio

R/ Es una unidad de código empaquetada, y se caracteriza por permitir ser invocada desde una petición http o https, permite exponer las funcionalidades sobre una unidad de red.

B. SOAP vs REST

R/ Son dos estándares diferentes, para la construcción y publicación de servicios.

C. Métodos GET, POST, PUT, DELETE: Definiciones.

R/ GET

El método GET se utiliza para solicitar datos de un recurso específico. Es el método más común y se usa para recuperar información. Con GET, los datos de la solicitud se envían en la URL.

POST

POST se utiliza para enviar datos al servidor para crear o actualizar un recurso. Los datos se envían en el cuerpo de la solicitud HTTP, no en la URL.

PUT

PUT se utiliza para enviar datos al servidor para crear un nuevo recurso o reemplazar una representación del recurso de destino con los datos de la solicitud. Los datos se envían en el cuerpo de la solicitud.

DELETE

DELETE se utiliza para eliminar un recurso específico.

D. Herramientas de Ejecución de Servicios: Experiencias.

R/ SoapUI y Postman, la herramienta que prefiero a pesar de SoapUI, ser una herramienta muy completa, es postman, por su flexibilidad para en esta realizar automatizaciones, configuraciones e importación de servicios, tiene mas herramientas que permitan agilizar la realización de verificaciones.

E. Automatización de Servicios: Proceso.

R/

- Diseñar los escenarios de prueba, lo más posibles cercanos a las definiciones de negocio, lo menos técnico posible.
- Procurar que los escenarios sean lo más completos posibles y no probar únicamente las solicitudes exitosas, sino los casos secundarios como errores de negocio y errores técnico (ser muy claros en estas especificaciones)
- Escoger la herramienta óptima para la automatización, lo más efectivo es una herramienta con que el equipo este familiarizado, para no tener esfuerzos aislados.
- Realizar la automatización.
- Realizar entrega al equipo de la automatización realizada

F. Validaciones en Automatización de Servicios: Ejemplos.

R/

- Enviar en la petición del servicio parámetros no esperados.
- Revisar que el servicio automatizado, si haya quedado con los estándares adecuados, es decir que en realidad correspondan a un GET, POST, PUT.
- Enviar en los cuerpos de la petición valores no esperados.

9. Pruebas No Funcionales

A. Seguridad y Performance: Explicación.

R/

- Pruebas de performance: son un conjunto de pruebas diseñados con ciertos criterios con el objetivo de brindarle al usuario disponibilidad a los servicios, y al negocio garantizarle que cuente con la resistencia requerida para suplir las necesidades del mercado.
- Pruebas de seguridad: conjunto de pruebas diseñadas, para brindar al usuario seguridad ante ciertos comportamientos maliciosos, ejecutados con la necesidad de vulnerar los recursos y procesos de las organizaciones.

B. Herramientas para Pruebas de Seguridad y Performance: Experiencias.

R/ He trabajado con Jmeter, realizando los respectivos mapeos de las peticiones y realizando un análisis del comportamiento de la infraestructura del aplicativo, ante pruebas de estrés y de carga, pero la experiencia no ha sido amplia

Y en seguridad no tengo ninguna experiencia.

C. Pruebas de Carga: Uso de herramientas como JMETER, WAPT, o BlazeMeter.

R/ Como lo comenté en la respuesta anterior uso de JMETER.

10. Automatización de Móviles

R/ He automatizado para aplicaciones nativas , en Android utilizando appium con el patrón de diseño Page Object y page factory.

A. Apps Nativas vs Híbridas y Sistemas Operativos: Descripción.

R/ La diferencia es que las nativas necesitan un instalador y un sistema operativo especializado para su ejecución.

Las hibridas están alojadas directamente en la nube requiriendo un browser para su ejecución y estando disponibles para diferentes sistemas operativos, teniendo en cuenta el tema de las resoluciones

B. Diferencias entre Automatización Web (Selenium) y Móvil (Appium)

R/ Radica en la configuración de los proyectos, radica de mayores recursos de máquina en appium para el montaje del emulador y sincronización con el framework.

Dado esto se cuenta con más posibilidades de automatización y de herramientas en cuanto la automatización web con selenium.

C. Granjas de Dispositivos Móviles: Experiencias.

R/ Trabaje con lambdatest automatizando en una ocasión con una aplicación hibrida, se debe de tener cuidado es la configuración inicial, pero una vez se completa los scripts de automatización siguen siendo los mismos que para una aplicación web corriente.

11. Seguridad

A. Conceptos de Seguridad en Pruebas Web y Móvil

R/ Esta área no he tenido experiencia ni conocimiento en conceptos.

B. Herramientas para Pruebas de Seguridad: Experiencias.

R/N/A

12. Tecnología Cloud

A. Experiencia con Proyectos Cloud y Herramientas Utilizadas

R/ Experiencia en las diferentes herramientas de aws, como cloudwatch para la revisión y verificación de comportamientos de lambdas, también verificando base de datos no sql en esta nube, con dynamodb, todo esto desde las pruebas funcionales.

13. Integración Continua

A. Tareas de Integración y Despliegue Continuo: Rol del Automatizador.

R/ automatización de pipelines con gitaction, azure, y Jenkins.

Con Jenkins tuve la oportunidad de automatizar un proceso completo de integración continua con un Linux Ubuntu server, ejecutando headless, en los flujos de ejecución de pruebas.

B. GIT: Concepto, comandos más usados, y experiencia con GitLab o GitHub.

R/ En git puedo mencionar:

- Uso de los pull request, como mecanismo par de validar el código de los compañeros de trabajo antes de autorizar un paso a la rama principal o secundaria del proyecto según corresponde.
- Manejo de los arhivos gitignore, para no subir archivos que solo son de uso en local, y pueden afectar en ejecución y almacenamiento en los pipeline.
- Manejo de ramas correspondiente a cada feature o fix, para mantener un flujo de trabajo adecuado en el proyecto.
- Realización add to index, para preparar los archivos para ser agregados a un commit.
- Fetch, para actualización de los estados de las ramas remotas.
- Pull, para actualización de las ramas remotas en ramas locales
- Push, para enviar nuestro código en la rama correspondiente, del respositorio remoto.

He tenido mas profundidad con GitHub pero con gitlab también he tenido experiencias.

C. Proceso de Integración Continua con Jenkins

R/ Si he tenido acercamiento con esta tecnología y hasta he tenido la posibilidad de realizar configuraciones preliminares, para la integración con el sistema operativo correspondiente.

También he trabajado con la creación y mantenimiento de pipelines (jenkinsfile)

D. Proceso de Integración Continua con Azure DevOps

R/ Con azure devops en la creación y mantenimiento de pipelines para la ejecución de automatizaciones.