**PRUEBA TECNICA QA AUTOMATION**

1. ¿Cuál sería el proceso ideal al principio de un sprint para determinar qué probar y cómo hacerlo?

Para determinar que probar y como hacerlo, lo mas ideal es verificar con el equipo ¿Qué va/debe ser entregado en el incremento resultante del sprint? En base a esto, se puede establecer el que y el como y en cuanto, teniendo en cuenta el mínimo producto viable.

1. ¿Cómo planea escribir casos de prueba y extraer la información de las historias de usuario?

Lo mas breve, concisos y descriptivos, extraer lo mas puntual de la HU y que tenga valor, ya que a veces se tienen formatos, lineamientos o demás que provocan tardanza al momento de crearlos y al ejecutarlos no se entiende la finalidad del mismo, entonces en mi opinión debería ser algo así:

* Titulo: Descripción corta pero detallada del caso.
* Paso a Paso.
* Dependencia(En caso de tenerla o si tiene precondición).
* Resultado Esperado.
* Microservicios: Me parece importante este ítem, ya que a veces no se tiene trazabilidad de donde o porque se utilizan dichos microservicios y si tal vez mas adelante se realiza un migración, esta información es de valiosa ayuda.

1. ¿Cómo manejaría las nuevas prioridades relacionadas con los errores críticos?

¿Resolver un Bug o desarrollar una funcionalidad nueva? Probablemente es lo que puede llegar a suceder al querer priorizar, se puede dar que se tome como un requerimiento con valor negativo, entonces los bugs se priorizan teniendo en cuenta el costo de no solucionarlos, es decir, hacer una ponderación de que implicación tiene el bug en el sistema, si impide culminar una transacción, si la experiencia de usuario es negativa, si el usuario final podría no elegir el producto la próxima vez, si este puede representar una falla de seguridad o mas, el saber priorizar es de gran ayuda ya que así se puede mitigar la deuda técnica.

1. ¿Cómo se informa de los problemas a los desarrolladores?

Normalmente se tienden a crear células agiles o scrum teams, que son equipos los cuales están compuestos por distintos roles, dev, scrum, producto owner. Al ser equipos autogestionables los defectos o bugs son dados a conocer hablando con el dev, posterior al equipo o mediante algún medio que se tenga contacto directo, ya que de esta manera la comunicación es mas efectiva y se pueden tomar decisiones de forma inmediata, posteriormente de acuerdo con los lineamientos se reporta en alguna herramienta de gestión de defectos. Caso tal que no se informe de inmediato, en la daily se deben mencionar.

1. ¿Cómo es compatible con los desarrolladores para escribir pruebas automatizadas?

Un poco confusa la pregunta, considero que la persona de automation, debe tener muy buenas competencias tecnicas para lograr ese empalme con los dev, a nivel de tecnologias existen diversas implementaciones, frameworks que permiten lograr esa intercomunicación.

1. Teniendo en cuenta que a una aplicación le faltan pruebas automatizadas, ¿cómo determinaría qué parte de la aplicación requiere automatización primero?

Se debe evaluar en primera instancia la madurez de los componentes de la aplicación ya sea frontend, backend, se puede dar el caso que alguno de los dos aun se encuentre en una etapa muy temprana donde sufran cambios constantemente y la automatizacion no tendria el valor requerido, independiente de esto, de acuerdo con la piramide de cohn, por buenas practicas deberíamos empezar con unitarias (normalmente entregadas por dev para continuar con qa) , seguidas por API o la capa de integración y por ultimo nuestra GUI. Por otra parte, si se denota que en las pruebas manuales son muy repetitivas , complejas o consumen mucho tiempo es buen indicio por donde iniciar.

1. Si una aplicación presenta problemas de rendimiento, lo que resulta en un tiempo de respuesta alto en algunas situaciones, ¿cómo puede ayudar al equipo a manejar este asunto?

Verificar en primera instancia la arquitectura de la aplicación, en mi opinión es muy importante ya que de esta manera se pueden descartar factores que pueden afectar, ejemplo, si tenemos un servicio u componente externo el cual no depende de nosotros y no tenemos legado y este presenta demasiada latencia, se cae o diversas cosas. Internamente, evaluar si los recursos(infraestructura) que se planificaron para el funcionamiento de la aplicación son suficientes respecto al trafico/uso que se esta recibiendo en la aplicación, se puede verificar replicando el comportamiento de forma cercana mediante pruebas performance, donde se puedan mitigar los diferentes escenarios, como se comporta si realizo carga, estress, picos. Importante la retroalimentación ya que de esta manera se puede establecer las acciones a tomar, ejemplo: Se tiene que escalar vertical u horizontalmente.

1. ¿Cómo ve su papel durante las fases de aseo y planificación de los trabajos pendientes?

No entiendo muy bien la pregunta, pero trato de relacionarla con la etapa de refinamiento, aportar de acuerdo a la visión del producto y resultados de los sprints anteriores, también sobre cuales deberían ser los incrementos de los futuros sprints mitigando posibles incovenientes en las HU.