Escenario 1 – E-commerce en riesgo:

Tu equipo ha estado probando un nuevo sitio de e-commerce regional con integración de pasarela de pagos y módulo de promociones. El lanzamiento está programado para el próximo lunes, pero se presentan los siguientes retos:

- El checkout móvil falla intermitentemente y no hay claridad si es problema de backend o del API de pagos.
- La automatización de regresión (Playwright) está arrojando falsos negativos desde el último merge.
- El cliente principal está molesto y amenaza con escalar el proyecto a la dirección si no se garantiza la fecha.
- Desarrollo afirma que “no se tocó esa parte” y pide liberar igual.
- El PM te presiona para dar una respuesta definitiva en menos de 3 horas.
- Dos QA del equipo están desmotivados y expresan que no sienten respaldo del área técnica.

Preguntas:

a. ¿Qué medidas tomarías para identificar la causa raíz del problema y decidir si liberar o no el proyecto?
b. ¿Cómo manejarías la presión del cliente y del PM mientras mantienes la objetividad en la decisión?
c. ¿Qué acciones implementarías para recuperar la confianza del equipo QA y reforzar su autonomía técnica?

Respuestas:

a. Reunión de crisis inmediata con QA, Dev y PM con el fin de ver el panorama completo, alinearnos todos al mismo objetvo y definir prioridades.

   Solicito revisión de logs y trazas del checkout móvil para diferenciar si es error de backend o pasarela, realizar pruebas manuales inmediatas en diferentes dispositivos, pruebas de API para garantizar conectividad.

   Solicito aislar falla en entorno controlado con test manuales y automáticos de regresión.

   Ejecutar pruebas de smoke regresivo manual más ejecución limpia de automatización para revisar la causa de falsos negativos, ya sea por dependencias, errores en código o falla en los pipelines.

   Defino el go, no go basado en riesgo, si afecta flujo de pago no se libera. Si solo es intermitencia no crítica, se libera con plan de contingencia y monitoreo.

b. Comunicó inmediatamente a todos los stakeholders para mantener un proceso profesional, transparente y que son eventos que pueden ser controlados:

- (Estamos ejecutando pruebas de aislamiento para confirmar si el error proviene del API externo, desarrollo o falsos negativos de QA, envío avances mañana y tarde.)

Dado el caso que el problema sea mayor y requiere desarrollo y pruebas adicionales, no comprometo la calidad, por lo cual comienzo a negociar con el cliente para priorizar entregas parciales y/o soluciones temporales para garantizar la continuidad del negocio sin afectar ingresos.
Finalmente presento una matriz de riesgo vs impacto para que PM y cliente vean la decisión con fundamentos.

c. Convocó reunión de respaldo y alineación con la finalidad de dejar claro que la decisión técnica está sustentada y mi equipo QA tiene voz. Adicional implementó la revisión de código QA antes de los despliegues.

- Implemento sesiones semanales con el fin de unir al equipo, dar feedback, celebrar logros y generar lecciones aprendidas tipo review del sprint.
- Realizó chequeo 1 a 1 con el fin de ver más allá de lo laboral, a veces el problema no es de trabajo, es a nivel personal.

Escenario 2 – Campaña con múltiples errores

Simultáneamente, otro proyecto enfrenta una crisis diferente:
Una landing de campaña fue liberada hace tres días y empezaron a llegar reportes de errores visuales y formularios que no envían datos. El cliente exige una solución inmediata, pero:

- No existe evidencia clara de los casos probados antes del release.
- El ambiente de staging no coincidía con el código desplegado.
- Marketing hizo cambios directos en el CMS sin notificar al equipo de QA.
- El líder de UX culpa a QA por “no detectar errores evidentes”.
- El equipo de desarrollo indica que el bug viene de una librería externa.

Preguntas:

a. ¿Qué plan de contención implementarías para resolver la crisis y recuperar la confianza del cliente?
b. ¿Qué acciones concretas tomarías con los equipos de UX, desarrollo y marketing para evitar que se repita?
c. ¿Qué lecciones aplicarías al proceso de QA para mejorar control de versiones, ambientes y trazabilidad?

Respuestas:

a. Congelar el ambiente productivo, dependiendo del tipo de despliegue solicitó hotfix branch o rollback manual según sea el caso. Posterior verifico versión desplegada y parámetros de commit para identificar el responsable.

Solicito pruebas exploratorias rápidas tanto visual como funcional para confirmar el alcance de errores.
Comunicó al cliente un plan claro, tipo: Detectamos inconsistencias de versión, corregimos en staging y desplegamos hotfix en 3 horas. En paralelo reforzamos el control de CMS para garantizar que no se presenten casos de este tipo nuevamente.

b. - Implemento flujos de aprobación QA previos a publicación.
   - Defino un Content Freeze antes de cada release.
   - Marketing solo puede publicar mediante pull request si y sólo si, hay una previa certificación de mi equipo QA.
   - Realizó un taller conjunto de responsabilidades y límites donde defino los alcances y límites de cada equipo para mantener una cultura estandarizada en todo el ciclo de vida del software.

c. - Estandarizar un control de versiones en CMS (Git/branching).
   - Establecer entornos sincronizados (Dev–Staging–Prod) con el fin de mantener los desarrollos y pruebas alineados evitando falsos positivos.
   - Automatizar verificación post despliegue para garantizar monitoreo y continuidad del negocio.
   - Implementar checklist de pre-release + registro de evidencia garantizando una certificación QA con resultado exitoso.

BLOQUE B – PROPUESTA TÉCNICA QA: ESTRATEGIA DE CALIDAD PARA PLATAFORMA MULTITENANT

Flare planea desarrollar una plataforma SaaS multitenant para agencias de publicidad. Cada agencia (tenant) tendrá su propio espacio para gestionar campañas creativas, piezas gráficas, versiones y flujos de aprobación.

La solución incluirá:

- Roles y permisos por tenant.
- Versionado y control de assets.
- Integraciones mediante API pública con sistemas externos.
- Procesamiento asíncrono de assets en segundo plano.
- Infraestructura AWS enfocada en seguridad, aislamiento de datos y rendimiento estable.

El proyecto debe estar preparado para CI/CD desde el inicio, con una proyección de 100 tenants activos y más de 50 000 assets mensuales.

Parte 1 – Estrategia QA para el proyecto

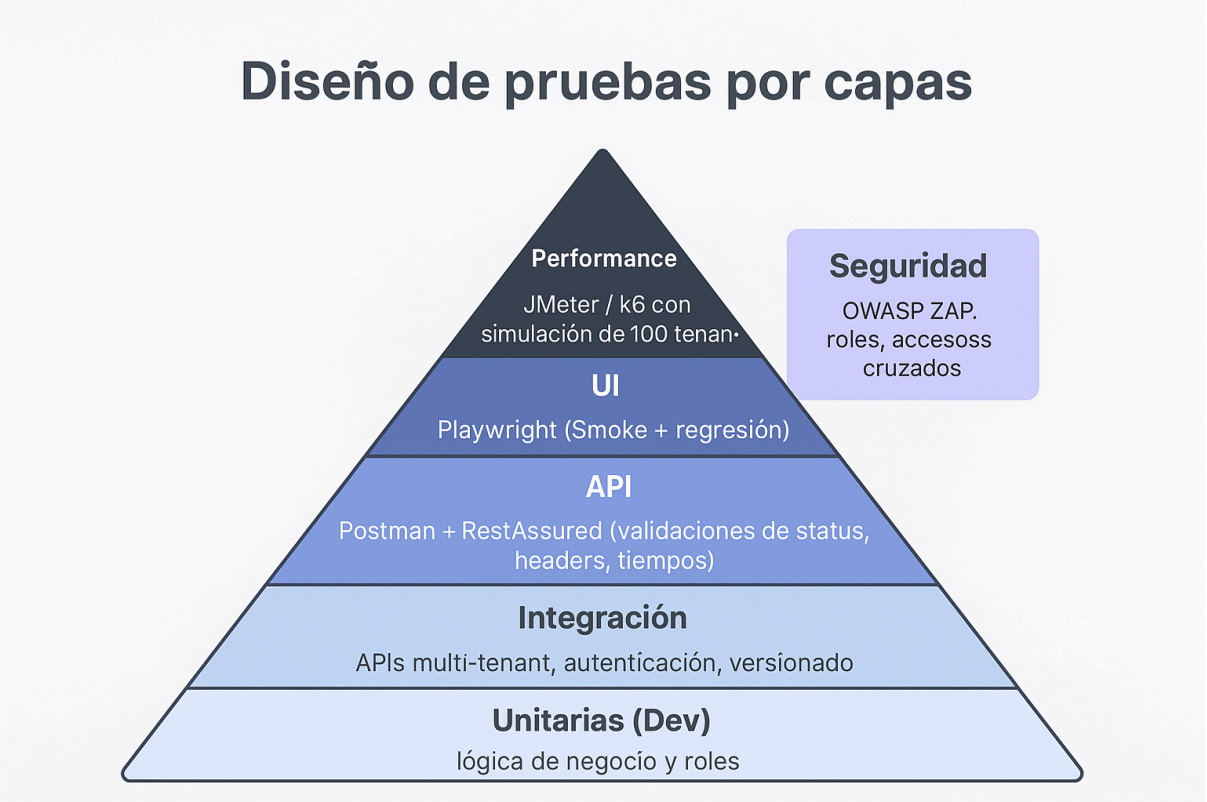
Define cómo estructuramos el aseguramiento de calidad de esta plataforma desde el día uno. El objetivo es demostrar visión, criterio técnico y pensamiento estructurado.

Incluye:

1. Visión general del enfoque QA. Cómo se integra QA al proceso desde la planificación hasta la entrega:

- QA se integra desde el sprint 0 con el fin de tener claro el proyecto desde su etapa de definición, planificación, ciclo de vida de desarrollo, implementación y entrega.
- Definición de criterios compartidos por parte del PO, el cliente, desarrollo y QA.
- QA participa en refinamiento y revisión técnica.
- Estrategia basada en pirámide de pruebas + CI/CD desde el inicio.

2. Diseño de pruebas por capas. Qué tipos de pruebas priorizarías (API, regresión, integración, accesibilidad, performance, etc.).



3. Gestión de entornos y datos. Cómo garantizarías consistencia y aislamiento entre tenants:
    - Manejo de un tenant por ambiente (Dev, QA, UAT para garantizar las aceptación del usuario final).
    - Pruebas multidispositivo y multinavegadores en paralelo para garantizar manejo de datos y sepracion de clientes y roles.
    - Datos anónimos o genéricos de pruebas y scripts de reseteo a la BD o manejo de reseteo manual.
    - Control de configuración por variables de entorno.

4. Automatización. Qué flujos automatizarías primero y por qué.

- Priorizaria flujos repetitivos y genéricos de los tenant, tales como login, formularios de registro, CRUD de Campañas, Assets y flujos de aprobación, adicional mediante pruebas exploratorias defino más flujos posibles para garantizar alta cobertura de automatización mediante playwright integrado a CI/CD ya sea git, ASW o azure dependiendo de la disponibilidad de plataformas.

5. Criterios de aceptación y cobertura. Cómo definirías el “done” junto al PM y desarrollo.

   - Código revisado.
   - Pruebas unitarias exitosas.
   - Pruebas de automatización de api e Integración. Exitosas.
   - Pruebas manuales de aceptación (según criterios definidos con cliente y PO) exitosas.
   - Pruebas de rendimiento/seguridad exitosas.
   - Documentación actualizada evidencias y logs de pruebas.

   Cobertura: Medir la cobertura de casos de prueba automatizados (>=80%).

6. CI/CD y métricas. Qué prácticas implementarías para integrar QA en los despliegues y medir la calidad:
   Implementaria puntos de validación en el pipeline para garantizar una ejecución punta a punta, ejemplo:

   De esta forma puedo definir KPI´s:
   Tiempo de ejecución del pipeline completo (<15 min).
   Calidad: porcentaje de builds exitosos, defectos encontrados en staging/prod.
   Eficiencia de QA: porcentaje de automatización de regresión, tasa de fallos de los tests <2%.

7. Cultura de calidad. Cómo fomentarías colaboración con Desarrollo, UX y Gestión para prevenir errores:

   Generando espacios mediante las ceremonias del sprint con desarrollo, QA y PO para realizar la planeación, los seguimientos diarios, refinar historias, documentar lecciones aprendidas y generar colaboración de todos los implicados.

   Implementando sesiones periódicas donde todo el equipo incluidos UX y stakeholders prueban el producto en busca de errores.

   Muestras de Calidad: Mi equipo QA presenta en demos los nuevos frameworks de automatización o las métricas de calidad de los proyectos en curso.

Parte 2 – Ejercicio práctico en stage real

Se te compartirá un entorno de pruebas (stage) disponible en https://greentex.quadi.io/.
Tu tarea es evaluar este sitio como QA Lead, identificando flujos relevantes —por ejemplo, formularios o procesos de envío de información— y demostrando cómo plantearías y automatizarías parte de las validaciones sobre este entorno.
El ejercicio no implica modificar el sitio ni generar tráfico alto; es únicamente demostrativo y confidencial.
Tu entrega debe incluir un análisis de los flujos críticos, una pequeña colección de pruebas en Postman con casos válidos y negativos, validaciones de campos y respuestas, una breve matriz de casos y los criterios de aceptación de al menos tres historias o flujos del sitio.

Agrega además una explicación corta (README) con los pasos para ejecutar la colección y una evidencia visual de los resultados, como una captura o export del resultado de ejecución en Postman donde se vean los tiempos de respuesta y los casos pasados o fallidos.

Se adjuntan resultados como archivos.