TRABAJO INTEGRADOR

Módulo 4

Versión 1.0

Natacha Rodriguez

Contenido

1. Consignas del proyecto

Este trabajo integrador se va a dividir en dos secciones, una que engloba a videojuegos e IA, y la otra unirá todo lo restante sobre procesos, mejoras y metodologías. En ambos casos, las resoluciones tendrán que ser una descripción lo más detallada posible, donde expliquen con sus propias palabras todo lo aprendido durante la cursada de este módulo.

1. Usted acaba de ingresar como gerente del área de calidad de software a una empresa que está en el mercado hace ya varios años. El producto es una empresa de streaming de documentales, que posee aplicaciones tanto en web como mobile, smart tvs, tablets, consolas, y cualquier otro dispositivo que haya en el mercado. Como gerente, dirige dos áreas de calidad, una manual y otra

de automatización, cada una relacionada con la otra, pero funcionan como células independientes. Después de una semana de trabajo, de conocer a los miembros del equipo, se elevaron ya varios comentarios de distintos tipos de problemas con procesos que hay en los equipos. Al recibir todos estos comentarios, decide sentarse con el CTO para discutir los límites de responsabilidades. “Confiamos plenamente, toma las decisiones necesarias”. Con toda esta información brindada realizar descripciones de cómo aplicaría procesos ágiles a los distintos equipos, y como las técnicas de prueba y de desarrollo ágil deberían ser implementadas. Realiza las gestiones de procesos y pruebas correspondientes. Identifica los procesos de automatización de pruebas e implementación. Planear las mejoras de procesos a futuro. Y por último describir y dar ejemplos de cómo llevar adelante la gestión de todo el proyecto a nivel calidad.

2. Usted acaba de ingresar a Guerrilla Games como tester de videojuegos y como parte del entrenamiento, las primeras semanas aprenderás sobre distintas formas de trabajar de la empresa, pero realizando testing en un juego ya presentado hace unos años. Project Zero Dawn era el nombre del proyecto, lo que hoy en día ya conocemos con Horizon Zero Dawn. Este videojuego tiene dos grandes cosas que la compañía quiere que investigues y armes una documentación explicativa de esto, el testing general (audio, controles, gráficos, localización, etc...) y la Inteligencia Artificial de los enemigos. Para ambas cosas, deberás explicar (podes agregar capturas de pantalla) con todo lo aprendido de ambas cosas, teniendo en cuenta detalles como cómo llevar adelante pruebas en las distintas áreas, que información captas al ver el videojuego funcionando, realizar investigaciones online de cómo funciona la IA en estos enemigos y como la probarías. Toda información que creas relevante para el área de testing sobre lo visto de este videojuego, tiene que ser detallada en el documento. Para las pruebas sobre Horizon Zero Dawn, entendemos que no todos pueden acceder a ese videojuego, así que con realizar la investigación del videojuego en youtube sobre esto, alcanza. Ver demos, playtesting de otras personas o transmisiones de gente jugando, todo esto alcanza y sobra para la magnitud del proyecto. Para la sección de IA de este videojuego, investigar en internet qué información pueden aprender sobre esto que no se ve necesariamente jugando o viendo videos.

Para realizar la entrega del trabajo, deberán crear una carpeta con su nombre acá, y dentro de la misma, cargar todos los archivos que crean necesarios para la aprobación. Recuerden que es obligatorio aprobar este trabajo para recibir el diploma de la carrera. El tiempo máximo de entrega es de dos semanas, con una posible re-entrega en caso de que el trabajo necesite correcciones para aprobar. En caso de dudas o consultas, dirigirse a los canales de Discord correspondientes o utilizar el espacio de consultas ofrecido por las tutoras.

Sección 1: Plan de Transformación Ágil (Caso Empresa de Streaming)

1. Introducción

El presente documento detalla el Plan Estratégico de Transformación de Calidad, diseñado para migrar nuestras operaciones hacia una cultura y metodología Ágil. Con el respaldo de la CTO - Dirección Técnica, este plan propone abandonar el enfoque reactivo donde la calidad es una responsabilidad compartida y transversal.

A lo largo de este informe, se definirán las estrategias para la implementación de marcos de trabajo, la adopción de prácticas y la aplicación de estrategias del equipo QA. Asimismo, se establecerán los criterios de gestión de riesgos, la arquitectura de automatización y los indicadores de mejora continua necesarios para garantizar que nuestra plataforma DocuMentales no solo funcione, sino que brinde una experiencia impecable a nuestros usuarios.

1.1 Contexto

Tras mi primera semana, he realizado un análisis profundo de la situación de DocuMentales, con más de 10 años en el mercado. Líder en documentales, pero perdiendo cuota en el mercado frente a gigantes como Netflix o Disney+. El equipo QA consta de 15 personas, 12 manuales y 3 de automatización.

Nuestros procesos de calidad están estancados en prácticas tradicionales que no soportan la velocidad del mercado de streaming.

He identificado los siguientes escenarios críticos en cada área, los cuales justifican las decisiones que detallaré más adelante:

1.1..1 Situación de los equipos:

Actualmente, el equipo de QA manual trabaja aislado del equipo de automatización. Hay una gran falta de comunicación entre ambos equipos, los testers manuales trabajan hace mas de 6 años en la empresa y se comunican muy poco con los automatizadores quedando sin conocimientos del negocio y siguiendo directivas de los testers más antiguos.

El equipo de Automatización funciona como una "fábrica de scripts" externa. Automatizan casos de prueba viejos que ya no son relevantes, y cuando encuentran errores, los reportan días después, cuando los desarrolladores ya están en otra tarea.

Hay una falta de colaboración y comunicación entre áreas técnicas y funcionales. La calidad se está "verificando" al final, en lugar de construirse desde el inicio.

1.1..2 Situación de las Pruebas

Utilizan Excel para documentar casos de pruebas. No hay trazabilidad.

Dada la fragmentación de dispositivos (Smart TVs, consolas, web, mobile), el equipo manual está saturado probando sólo las versiones Gold pocos días antes del lanzamiento.

Se están ejecutando regresiones manuales eternas que consumen el 80% del tiempo del sprint, dejando muy poco tiempo para pruebas exploratorias o de usabilidad.

1.1..3 Situación de la Automatización

La estrategia de automatización actual está invertida. Se han enfocado en automatizar el 80% de las pruebas a nivel de interfaz de usuario, que son lentas y frágiles, y hay unos pocos flujos sin mantenimiento de API testing.

Es muy frecuente que ante cambios en el diseño se rompen tests automáticos. El equipo de automatización pasa más tiempo arreglando scripts viejos que creando valor nuevo.

1.1..4 Situación del producto y procesos

Actualmente informan que aplican scrum, pero ahondando en las practicas es mas un waterfall en sprints. Los testers manuales reciben las versiones de desarrollo cuando completan el 100% de los criterios de aceptación de la historia de usuario, que se podrían clasificar como épicas, y recién pasa al entorno de testing, lo que genera un cuello de botella antes del lanzamiento.

Han elevado múltiples reportes sobre problemas de comunicación y duplicidad de tareas. El modelo actual de "pasamanos" entre equipos retrasa el Time-to-Market.

Diagnóstico: La organización requiere abandonar la segregación de tareas para adoptar un modelo de Agilidad Organizacional que integre la calidad en el origen del desarrollo

1.2 Propósitos

El propósito fundamental de este plan es reestructurar el área de Calidad para abandonar el modelo actual de "silos independientes" (manual vs. automatización) y transicionar hacia un modelo de **Agilidad Organizacional**. El objetivo es alinear la calidad con la velocidad de entrega que exige el mercado de streaming, eliminando los cuellos de botella generados por la falta de comunicación y la detección tardía de errores.

Este plan describe la transición hacia metodologías ágiles en este caso se aplica **Scrum**, la implementación de integración continua (**CI**) y la adopción de una estrategia de pruebas basada en riesgos.

Modernizar la infraestructura para permitir entregas rápidas (Time-to-Market). El objetivo que la empresa pueda liberar versiones sin riesgo. Priorizarás la construcción de pipelines de Integración Continua (CI/CD), la automatización de APIs para reducir tiempos de regresión y la implementación de métricas técnicas como Lead Time y Defect Removal Efficiency. Ya que se cuenta con automatistas especializados de gran experiencia, el objetivo es que el equipo de automatización sea el motor que permite sostener la velocidad de entrega en entornos ágiles sin comprometer la calidad.

En cuanto a QA manual, priorizaremos el testing exploratorio de los testers senior, las pruebas de usabilidad y rendimiento, y alinear las métricas de QA con los objetivos de negocio. La automatización será solo un soporte para liberar tiempo para pruebas manuales de alto valor.

1.3 Descripción

Actualmente, el área de calidad enfrenta problemas de comunicación, duplicidad de esfuerzos y detección tardía de defectos debido a la separación física y lógica entre testers manuales y automatizadores. Este plan describe la transición hacia metodologías ágiles en este caso se aplica **Scrum**, la implementación de integración continua (**CI**) y la adopción de una estrategia de pruebas basada en riesgos.

La organización se encuentra en un estadio de madurez inicial, con procesos fragmentados. La propuesta consiste en implementar el **enfoque de equipo completo** vista en los fundamentos de desarrollo ágil. Esto implica que la calidad dejará de ser una fase final y aislada para convertirse en una responsabilidad compartida e integrada en el ciclo de desarrollo desde el inicio.

No puedes cambiar todo de un día para el otro, un "Big Bang" porque el equipo se resistiría y el desarrollo se frenaría. Se propone plantear una estrategia de transición en 3 fases (corto, mediano y largo plazo). Con un criterio más realista y con visión estratégica que entiende la complejidad de la gestión del cambio.

Fase 1: Estabilización y estandarización

El objetivo es detener el caos y establecer un lenguaje común.

1. Unificación de herramientas: migrar todos los reportes (Excel, correos) a un flujo único en Jira. Nadie debe gestionar bugs por fuera del sistema.
2. Definición de reglas del juego – Gatekeeping: Establecer y documentar los criterios de entrada y salida (). En la sección 6 de este documento se amplían tales criterios.
3. Cross-Training (Capacitación Cruzada): Iniciar sesiones o llamadas grupales donde los QAs de Automatización enseñen a los Manuales a ejecutar scripts básicos, y los Manuales enseñen a los Automatizadores sobre reglas de negocio complejas. En la sección 4.1 se aplia el plan de capacitación propuesto para esta etapa 1.

4. Rituales Conjuntos: Se establece la obligatoriedad de la participación de QA en las Daily Scrums de los equipos de desarrollo. Busca eliminar el 'teléfono descompuesto' donde QA se entera de los cambios de diseño o negocio días después de que ocurrieron. Estas reuniones dejarán de ser técnicas exclusivas de programación para convertirse en puntos de sincronización del squad multidisciplinario, conformado por:
    - Product Owner / PM: Para visión de negocio y prioridades.
    - Desarrolladores: Para estado técnico.
    - UX/UI: Para validación de diseño y flujos.
    - QA (Manual y Auto): Para reportar estado de pruebas y bloquear/desbloquear historias.

Fase 2: Integración técnica y automatización

El objetivo es romper los silos y acelerar el feedback.

1. **Disolución de células**: Eliminar las áreas separadas de QA manuales y automatistas. Asignar QAs fijos a cada Squad de desarrollo (aplicación del modelo Whole Team Approach).
2. **Diseño e Implementación de Arquitectura gTAA:** Antes de escalar la automatización, se definirá la estructura base del framework separando la capa de definición (datos/Gherkin) de la capa de adaptación (drivers Selenium/Appium). El objetivo es garantizar que los scripts del piloto sean reutilizables y mantenibles.
3. **Proyecto piloto de automatización:**
    a) Implementación de la gTAA en un solo módulo crítico (ej. Pasarela de Pagos).
    b) Implementar el pipeline de CI/CD con Smoke Tests automáticos en un solo módulo crítico (Se detallan tales módulos en la sección 9) para demostrar el valor (Quick Win) antes de escalar. En este proyecto se ubica la gTAA en la Fase 2 porque coincide con el proyecto piloto. No se busca automatizar masivamente en la Fase 3 sobre una base desordenada. Se usa la Fase 2 para construir los cimientos sólidos, la arquitectura en capas, en este caso, y probarlos con el piloto, para luego escalar con seguridad.
4. **Implementación de BDD:** Comenzar a redactar las nuevas Historias de Usuario con formato Gherkin para alinear a Producto, Dev y QA (se explican tales criterios en la sección 3).

Fase 3: Optimización y aplicación del shift left testing

El objetivo es escalar la automatización sobre la arquitectura probada en la fase 2 y consolidar la cultura de prevención.

1. **Expansión de la automatización de regresión (Rollout):** Una vez validada la arquitectura gTAA en el piloto (fase 2), pasamos de automatizar solo el Smoke Test a cubrir la gran mayoría de los flujos de negocio para liberar al equipo manual.
    i. **Estrategia de Capas:** basándonos en la pirámide de pruebas
        A. Capa de API (60% del esfuerzo): Automatizaremos masivamente las pruebas de backend (microservicios de streaming, pasarela de pagos, gestión de usuarios). Son rápidas, estables y fáciles de mantener.
        B. Capa de UI (20% del esfuerzo): Automatizaremos solo los flujos End-to-End críticos (Registro -> Login -> Pagar -> Ver Video). Estas pruebas son frágiles y lentas, por lo que seremos selectivos.
    ii. **Mantenimiento y Estabilidad:** Estableceremos una política de "Cero Flaky Tests". Si una prueba falla aleatoriamente sin cambios en el código, se retira del pipeline inmediatamente para ser reparada, evitando que el equipo pierda confianza en la automatización.
    iii. **Ejecución Nocturna:** Implementación de una suite de regresión completa que se ejecuta cada noche. Al llegar a la mañana, el equipo tiene un reporte del estado de salud de toda la plataforma.

2. **Implementación del testing temprano (Shift Left**): QA participa activamente en el refinamiento de requisitos. El objetivo ya no es encontrar bugs, sino evitar que se escriban. Se busca reducir el costo de la no calidad, basándonos en el criterio de que corregir un error en la etapa de requisitos cuesta potencialmente menos que en producción.
    - Revisión estática de requisitos: los QA participan en las sesiones de refinamiento (*backlog grooming*) no como oyente, sino como auditor. product owner + desarrollador + QA (conocido como "los tres amigos") evalúan cada historia antes de que entre al sprint. QA valida que los criterios de aceptación sean verificables, no ambiguos y cubran los "casos negativos".
    - Diseño de pruebas antes del código (ATDD): implementación de *acceptance test driven development*[1]. El QA que corresponda escribe los escenarios de prueba (en Gherkin) antes de que el desarrollador escriba una sola línea de código. El desarrollador usa esos escenarios como guía, garantizando que el código cumpla exactamente con lo esperado desde el primer intento.

3. **Gestión de métricas avanzadas (KPIs de eficiencia):** Abandonamos las métricas de vanidad cómo número de casos ejecutados para centrarnos en métricas de valor que impulsan la mejora continua aplicando el ciclo PDCA[2].
    - **Monitoreo y Optimización de la Salud Arquitectónica (Performance de la Suite):** El éxito de la Fase 3 traerá consigo un aumento exponencial en la cantidad de scripts. Sin una gestión activa, el tiempo de ejecución crecerá linealmente hasta volverse inmanejable, rompiendo el principio de feedback rápido de la agilidad. Para evitar la degradación del pipeline, implementaremos las siguientes acciones sobre la Capa de Ejecución de la gTAA:
        1. Establecimiento de Umbrales (KPI de Tiempo): Se define un SLA (Acuerdo de Nivel de Servicio) interno para el equipo de QA: El Smoke Test no debe superar los 5 minutos y la regresión completa no debe superar los 60 minutos. Si se cruza este umbral, se bloquea la creación de nuevos scripts hasta optimizar la ejecución actual.
        2. Estrategia de Paralelización (escalado horizontal): abandonaremos la ejecución secuencial (un test detrás de otro). Configuraremos la Capa de Ejecución para distribuir los tests en múltiples nodos o contenedores simultáneos (ej. Selenium Grid ocontenedores Docker dinámicos en el pipeline). Ejecutar 500 tests distribuidos en 10 nodos paralelos reduce el tiempo total de 100 minutos a 10 minutos teóricos.
        3. Refactorización de "Wait Times": auditoría de código para eliminar esperas estáticas (Thread.sleep) y reemplazarlas por esperas explícitas inteligentes (WebDriverWait). Esto hace que el script espere "solo lo necesario" y no tiempos fijos, acelerando la ejecución global.
    - **DRE (Defect Removal Efficiency - Eficacia de eliminación de defectos):**

$$DRE = \frac{Defectos\ encontrados\ por\ QA}{Defectos\ por\ QA + defectos\ encontrados\ en\ producción}$$

[1] *ATDD - Acceptance Test Driven Development (desarrollo guiado por pruebas de aceptación) es convertir los requisitos del cliente en pruebas ejecutables antes de programar, para asegurar que todos entiendan lo mismo desde el principio.*
[2] *El ciclo PDCA (Plan – Do – Check – Act) en español Planificar, Hacer, Verificar, Actuar, metodología de gestión para la mejora continua de procesos.*

Objetivo: Mantener el DRE por encima del **95%**. Si baja, significa que nuestros filtros están fallando y debemos analizar la causa raíz en la Retrospectiva fase *Check o Verificar* del PDCA.

- **Lead Time y Cycle Time:** Mediremos cuánto tiempo tarda una Historia de Usuario desde que se crea hasta que está probada y lista para producción (*Lead Time*).

  Objetivo: Reducir este tiempo eliminando las esperas entre Dev y QA (desperdicio o *Waste* en Lean).

- **Densidad de Defectos:** Analizaremos qué módulos (ej. "Módulo de Pagos" vs "Módulo de Perfil") tienen más errores por línea de código para asignar a nuestros testers más experimentados (Seniors) a las áreas más problemáticas.

Esta fase 3 de optimización busca transformar el rol de QA, pasamos de ser 'detectives que buscan culpables' a ser 'consultores de calidad' que ayudan a construir el producto correctamente desde el principio. Al automatizar la regresión, lo repetitivo, liberamos el talento humano para hacer testing exploratorio, por ejemplo, en las nuevas funcionalidades, que es donde realmente aportamos valor creativo. Si nuestra regresión tarda 4 horas, los desarrolladores dejarán de correrla y volveremos a tener bugs en producción. La paralelización es la única forma de mantener el feedback loop corto a medida que agregamos más pruebas y que se complejiza la aplicación.

1.4 Objetivos estratégicos

- **Integración cultural (Whole Team Approach):**
  - Eliminar la segregación operativa entre "QA Manual" y "Automation".
  - Integrar a los profesionales de calidad dentro de **Squads Multidisciplinarios** para asegurar una responsabilidad compartida sobre el producto.
- **Nivelación de competencias (Perfiles T-Shaped):**
  - Implementar un programa de **Cross-Training** (Capacitación Cruzada) donde los testers manuales aprendan ejecución técnica y los automatizadores aprendan reglas de negocio para reducir la dependencia de especialistas y evitar cuellos de botella.
- **Cultura de prevención (de QC a QA):**
  - Transicionar de un enfoque reactivo de control de calidad – QC - enfocado en la detección de errores, a uno proactivo de aseguramiento de calidad - QA – Enfocado en la prevención de errores.
  - Aplicar Shift Left involucrando a QA en la definición de Historias de Usuario (ATDD) para evitar defectos desde el diseño.
- **Eficiencia Técnica (Arquitectura gTAA):**
  - Implementar una **Arquitectura de Automatización de Pruebas Genérica (gTAA)** escalable, separando datos de pruebas y scripts.
  - Enfocar la automatización en la regresión y APIs para liberar tiempo de testing manual exploratorio en dispositivos complejos.

2. Alcance y pruebas

Priorización basada en riesgo y valor de negocio. El objetivo es maximizar la cobertura en los flujos críticos, mientras se reduce el desperdicio en áreas de bajo impacto o tecnología obsoleta. Se prupone el alcance para las fases 1 y 2:

2.1 Alcance (In-Scope):

2.1..1 **Funcionalidades de Negocio (Core Business):** Lógica de negocio punta a punta, end to end en los siguientes módulos críticos***:***

1. **Módulo de identidad (Auth):**
   - Login/registro: Validación de credenciales propias y OAuth (Google/Facebook)
   - Gestión de sesión: Persistencia de la sesión entre dispositivos (ej. empezar en móvil, terminar en TV) y manejo de *tokens* de seguridad.
   - Restricción Geográfica: Verificación de bloqueo/acceso de contenido según IP del usuario (Licencias por país).
2. **Módulo de Monetización (Billing):**
   - **Pasarela de pagos:** Integración con proveedores actualmente Mercado Pago para el mercado local (pesos argentinos) y PayPal Directo + Nubi/Macro/Bitso para el mercado internacional (dólares). Se validarán escenarios de éxito, rechazo por fondos, y expiración de tarjetas.
   - **Ciclo de vida de suscripción:** *Upgrades* (de Plan Básico a 4K), *Downgrades* y Cancelaciones. El sistema no debe cortar el acceso inmediatamente tras una cancelación efectiva. Permitiendo al usuario usar el servicio hasta el ultimo día que tiene pagado.
3. **Módulo de Reproducción (Player):**
   - **Streaming Adaptativo:** Verificación de que el bitrate sube o baja según el ancho de banda simulado (Throttle testing).
   - **Recuperación de fallos:** Comportamiento del player cuando se corta internet y vuelve (debe reanudar, no reiniciar).
   - **DRM (derechos digitales):** Verificación de que el contenido no puede ser descargado ilegalmente.

2.1..2 **Cobertura de Plataformas (Matriz de Compatibilidad):** Dada la alta fragmentación del mercado de Smart TVs, acotamos el alcance a los dispositivos que representan el 85% de nuestra base de usuarios activa en función a las métricas actuales de la aplicación. Para adoptar una cobertura basada en riesgos y cuota de mercado. Como propuesta, hasta tener estos resultados, nos basamos en las tendencias globales.

La Estrategia de los "tres niveles" (Tiered Approach)

Para racionalizar la matriz, dividiremos el universo de dispositivos en tres niveles. Esta segmentación se debe basar en tus analíticas actuales (Google Analytics, Firebase, Mixpanel) y tendencias globales.

Tier 1: "Gold Standard" (Soporte Crítico)

- Cobertura: Representa el 70-80% de tu tráfico/ingresos.
- Acción: Automatización obligatoria (CI/CD) + Pruebas manuales exhaustivas en cada release.
- Dispositivos: Los "Flagships" actuales y la versión anterior inmediata.

Tier 2: "Silver Standard" (Soporte Importante)

- Cobertura: El siguiente 15-20% del tráfico.
- Acción: Pruebas manuales en "Smoke Tests" o regresiones parciales. Automatización básica (Happy paths).
- Dispositivos: Modelos de hace 2-3 años o marcas secundarias populares.

Tier 3: "Bronze Standard" (Best Effort)

- Cobertura: El "Long Tail" (dispositivos antiguos o muy nicho).

- Acción: Pruebas reactivas (solo si un usuario reporta un bug crítico) o Crowdsourced testing.
- Dispositivos: Sistemas operativos obsoletos pero funcionales, consolas antiguas.

Incorporamos pruebas de Ingeniería de Streaming:

- DRM (Gestión de Derechos Digitales): validación de desencriptado de licencias Widevine (Android/Web), FairPlay (Apple) y PlayReady (TVs).
  - Criterio de Aceptación: El video debe iniciar en menos de 2 segundos sin errores de licencia (Error 600x).
- ABR (Adaptive Bitrate Streaming):
  - Pruebas de Network Throttling: Simularemos caídas de ancho de banda (de WiFi 100mbps a 3G inestable) para asegurar que el player baja la calidad suavemente sin detenerse (buffering).
- Calidad de Imagen y Codecs:
  - Validación de reproducción en 4K, HDR y Dolby Vision en dispositivos Tier 1.
  - Pruebas de decodificación H.264 vs H.265 (HEVC) para asegurar compatibilidad en hardware antiguo (Tier 2).
- Localización Audiovisual (Subtítulos):
  - Verificación de cambio de audio/subtítulos "al vuelo" (sin recargar).
  - Sincronización labial (Lip-sync) y renderizado correcto de caracteres especiales en Smart TVs.

2.2 Fuera del Alcance (Out-Scope):

Para garantizar la viabilidad del plan y maximizar el ROI del equipo de QA, las siguientes áreas, dispositivos y tipos de prueba quedan explícitamente excluidos de este ciclo de trabajo. Cualquier defecto reportado en estos ámbitos será clasificado como "Won't Fix" (No se arreglará) o "Backlog Future" (Para el futuro), salvo excepción directa del Comité de Dirección.

1. **Dispositivos y plataformas obsoletas (Tier 3)**
   - Smart TVs Legacy: Modelos de Samsung Tizen y LG WebOS fabricados antes de 2019. El hardware de estos dispositivos no soporta los nuevos códecs de video eficientes (H.265) ni las librerías de seguridad actuales.
   - Consolas de Antigua Generación: PlayStation 3, Xbox 360 y Nintendo Wii. El tráfico proveniente de estas consolas es inferior al 0.5%.
   - Navegadores Descontinuados: Internet Explorer (cualquier versión) y versiones de Edge anteriores a Chromium.
   - Sistemas Operativos Móviles Antiguos: Android 9 o inferior, y iOS 14 o inferior.
2. **Automatización de Interfaz (UI) en sistemas legacy**
   - Monolito Java (Web Admin): No se crearán scripts de Selenium/Cypress para el sistema administrativo antiguo (Backoffice Legacy). Las pruebas en este sistema serán estrictamente manuales y limitadas al mantenimiento básico.
   - Automatización de UI en Smart TVs (Fase 1): Durante este trimestre, no automatizaremos la navegación visual en las TVs (debido a la complejidad y costo de herramientas como Suitest). Confiaremos en el testing exploratorio manual y en la automatización de la API subyacente.
3. **Pruebas no funcionales de gran escala:** Aunque críticas, estas pruebas requieren infraestructura y tiempos que exceden el alcance del equipo interno actual en esta fase de transformación.
   - Pruebas de Estrés Masivo (>100k usuarios): No simularemos escenarios de "Estreno Mundial" en este sprint. Nos limitaremos a pruebas de carga basal (Performance Baseline).

- Hacking Ético / Pentesting Profundo: La auditoría de seguridad avanzada será realizada por un proveedor externo certificado una vez al año. El equipo interno solo ejecutará análisis estático de seguridad (SAST) en el código.

4. **Defectos cosméticos menores**: No se aceptarán reportes de defectos por diferencias visuales menores (píxeles, alineación leve, fuentes) en dispositivos de Tier 3 o navegadores secundarios (Opera, Vivaldi), siempre que la funcionalidad principal (reproducir video) no se vea afectada.

3. Criterios de aceptación

Para garantizar que el desarrollo cumpla con las expectativas del negocio antes de iniciar las pruebas, utilizaremos la metodología **BDD (Behavior Driven Development)**. Cada Historia de Usuario deberá contar con criterios de aceptación redactados en lenguaje **Gherkin** (*Given-When-Then*).

Ejemplo:

- **GIVEN** que el usuario tiene una suscripción activa y ancho de banda > 10Mbps
- **WHEN** selecciona un documental en 4K
- **THEN** la reproducción inicia en menos de 3 segundos sin buffering.

4. Recursos y herramientas

4.1 Recursos humanos

Bajo el nuevo esquema de *Whole Team Approach*, los roles se redefinen:

- **QA Manager:** Responsable de la estrategia, métricas y gestión de riesgos.
- **QA Automation Engineers:** Integrados en los Squads, encargados de mantener el framework de automatización y el pipeline de CI.
- **QA Manual / Exploratory Testers:** Especialistas en UX y dispositivos complejos (TVs/Consolas), enfocados en encontrar defectos no triviales.
- **Desarrolladores:** Responsables de las pruebas unitarias y de colaborar en la automatización de integración (TDD).
- **IX/UI:**

Plan de capacitación

Para romper las barreras de conocimiento y reducir el conocido factor bus[3] implementaremos un programa bidireccional de transferencia de habilidades. El objetivo es evolucionar de especialistas aislados a profesionales generalistas-especialistas - T-Shaped[4]. Se organizaría en pequeñas reuniones semanales de 1 hora, 30 min de automatización a manual y 30 min de manual a automatización.

[3] Riesgo de que el conocimiento se concentre en una sola persona.
[4] Es experto en una cosa, pero versátil en otras. A diferencia del I-Shaped, Solo tiene la barra vertical. Es un experto en una sola cosa y no sabe nada de lo demás.

1. **Automatización a manual:** El objetivo es que los testers manuales ganen autonomía técnica y pierdan el miedo al código. Temas a tratar:
    a. **Ejecución de Pipelines:** Los ingenieros de automatización capacitarán al equipo manual sobre cómo lanzar las suites de pruebas (Smoke y Regresión) desde la herramienta de CI (ej. Jenkins/GitLab) sin necesidad de tocar código. El tester manual no tiene que esperar al automatizador para saber si la build está estable.
    b. **Análisis de Reportes y Logs:** cómo interpretar un fallo en la consola. Diferenciar un "Error de Script" (falso positivo) de un "Bug de Producto" real.
    c. **Mantenimiento Básico:** Capacitación en la identificación de objetos. Si cambia el ID de un botón y el test falla, el tester manual debería ser capaz de actualizar ese ID en el repositorio de objetos sin reescribir la lógica.
2. **Manual a automatización:** El objetivo es que los automatistas comprendan el valor de negocio y eviten de escribir scripts "ciegos" y que no aporten valor. Deben planificarse y evaluarse los test para que tengan un retorno de inversión considerable. Temas a tratar:
    a. Reglas de negocio complejas: Los testers manuales (que suelen ser los expertos en el dominio) explicarán los flujos críticos y las casuísticas "borde".
    b. Evita que se automaticen pruebas triviales que no aportan valor.
    c. Heurísticas de UX y dispositivos: Transferencia de conocimiento sobre las peculiaridades de cada plataforma o dispositivos.
    d. Testing Exploratorio: Sesiones conjuntas donde el automatizador observa cómo explora un tester manual para identificar nuevos escenarios candidatos a automatización que no estaban en los requisitos iniciales.

Es una propuesta inicial, de ser necesario se ajusta a las necesidades del equipo.

4.2 Herramientas de software

- **Gestión:** Jira (Trazabilidad de Defectos e Historias), Confluence (Documentación).
- **Automatización:** Selenium WebDriver (Web), Appium (Mobile/TV), Postman/RestAssured (API).
- BrowserStack o Sauce Labs.
- **CI/CD:** Jenkins o GitLab CI (Ejecución de pipelines).
- **No Funcionales:** JMeter (Performance), OWASP ZAP (Seguridad).

4.3 Herramientas de hardware

- Para las pruebas en Smart TVs y Consolas: **Montaje de un Device Lab** en la oficina con las 10 TVs más representativas del Tier 1 (Samsung, LG, Sony). Se toma esta decisión, ya que los emuladores no replican fielmente la experiencia de reproducción de video 4K ni la interacción con controles remotos.

5. Entornos

Se establece una política estricta de separación de ambientes:

1. **Desarrollo (DEV):** Entorno local o nube inestable donde los desarrolladores integran código. Se corren Pruebas Unitarias.
2. **Testing / Staging (QA):** Réplica lo más exacta posible de producción. Aquí se ejecutan los Smoke Tests, Regresión Automatizada y Pruebas Manuales. Los datos están anonimizados.

3. **Producción (PROD):** Entorno real. Se realiza monitoreo sintético y pruebas "Canary" controladas.

6. Criterios para realizar las pruebas

6.1 Criterios de entrada (definition of ready para QA)

QA comenzará la ejecución solo si:

1. Requisitos de negocio y funcionales:
- Formato: La historia sigue la estructura: "Como [Rol], quiero [Acción], para [Valor]".
- Criterios de aceptación (Gherkin): Debe incluir al menos 3 escenarios redactados en formato Dado / Cuando / Entonces (Escenario Feliz, Escenario Negativo y Caso Borde).
- Matriz de dispositivos definida: Se debe especificar explícitamente en qué Tier de dispositivos aplica este cambio.

2. Requisitos de experiencia de usuario (UX/UI)
- Mockups / prototipos: Enlace a Figma/Adobe XD con el diseño final aprobado.
- Comportamiento de navegación (Focus): Para Smart TVs, debe especificarse el orden del foco al usar las flechas del control remoto (Arriba/Abajo/Izquierda/Derecha).
- Estados de Error: Diseño visual de los mensajes de error
- Textos Finales (Copy): Los textos legales y de interfaz deben estar definidos por el área pertinente.

3. Requisitos técnicos:
- La Build es exitosa y ha pasado los Smoke Tests automatizados.
- El código está desplegado en el entorno de testing.
- Los datos para las pruebas deben estar disponibles y ser fiables.
- Para pruebas de servicios: contrato de API (Swagger/OpenAPI debe estar definido el endpoint: Método (GET, POST), Payload de entrada (JSON esperado), Códigos de respuesta (200, 400, 401, 500). Y los datos de prueba necesarios.
- Dependencias Externas: Si la historia depende de una API de terceros (ej. Mercado Pago, PayPal, etc.) las credenciales de prueba deben estar disponibles.

6.2 Criterios de salida (definition of done para QA)

Para considerar una tarea Done desde el punto de vista de calidad, debe cumplir estrictamente:

- Todos los escenarios Gherkin críticos (P1) automatizados en la capa de API.
- Ejecución de prueba exploratoria en al menos un dispositivo físico Tier 1 (TV).
- Cero defectos de Severidad Crítica o Alta abiertos.
- Código revisado y fusionado en la rama Main sin romper el build (CI en verde).

7. Riesgos

Adoptamos un enfoque de Risk-Based Testing (RBT) reconocemos que no podemos probarlo todo, por lo que asignamos nuestros recursos de calidad proporcionalmente al nivel de riesgo detectado.

A continuación, se detallan los riesgos críticos identificados para el ciclo de transformación actual y sus planes de acción.

7.1 Riesgos de producto:

Estos riesgos amenazan directamente la estabilidad de la plataforma y la experiencia del usuario final.

Riesgo Identificado	Probabilidad	Impacto	Estrategia de Mitigación	Plan de Contingencia
Inestabilidad del Legacy El sistema antiguo (Monolito) se rompe al interactuar con los nuevos microservicios.	Alta	Crítico	**Aislamiento**: Implementar pruebas de contrato (*Contract Testing*) entre el Legacy y los nuevos servicios para detectar rupturas de esquema inmediatamente.	**Rollback:** Procedimiento automatizado para revertir el despliegue a la versión estable anterior en <5 minutos.
Fragmentación de Smart TVs La app funciona en Samsung (Tier 1) pero falla en una TV marca "White Label" antigua (Tier 3).	Muy Alta	Medio	**Alcance acotado**: Aplicar estrategia de *Tiers*. Solo garantizamos soporte oficial en Tier 1 y 2. Tier 3 es "Best Effort".	Si la TV no soporta el player moderno, mostrar un mensaje amigable.
Falsos positivos en UI La automatización de UI falla constantemente por cambios estéticos, perdiendo confianza.	Alta	Alto	Pirámide de Pruebas: Mover el 80% de las validaciones a la capa de API. Automatizar UI solo con selectores robustos y solo en flujos críticos.	Ejecución Manual: Si la suite de UI falla, se ejecuta un smoke test manual rápido para desbloquear el release mientras se arregla el script.
Dependencias de terceros Caída o lentitud de la Pasarela de Pagos (PayPal/Stripe) durante las pruebas.	Media	Crítico	Service Virtualization: Uso de Mocks y Stubs para simular las respuestas de la pasarela de pagos, evitando depender de su Sandbox.	Switch de Pasarela: (en producción) El sistema debe poder ocultar la opción de pago caída y ofrecer la alternativa automáticamente.

7.2 Riesgo del proyecto

Estos riesgos amenazan la capacidad del equipo para entregar a tiempo y con calidad.

Riesgo Identificado	Probabilidad	Impacto	Estrategia de Mitigación (Preventiva)	Plan de Contingencia (Reactiva)
Resistencia al Cambio (Silos) Testers manuales y Automatizadores no colaboran, generando bloqueos.	Alta	Alto	Objetivos compartidos: Los OKRs de los squads premian la calidad grupal, no individual. Implementación de sesiones de Pair Testing obligatorias.	Intervención: Reasignación de miembros conflictivos y talleres de Team Building facilitados por Agile Coach.
Cuello de Botella en Hardware Escasez de dispositivos físicos (TVs) para probar antes del release.	Media	Alto	Reserva de Lab: Calendario estricto de uso del "Device Lab". Uso de granjas en la nube para todo lo que no sea TV.	Horas Extra / Turnos: Establecer turnos escalonados para maximizar el uso del hardware disponible (mañana/tarde).

Baja Cobertura de Unit Tests Devs entregan código sin pruebas unitarias, saturando a QA de bugs básicos.	Alta	Crítico	Definition of Ready (DoR): QA tiene la potestad de rechazar cualquier ticket que no tenga evidencia de Unit Tests aprobados.	Code Freeze: Detener el desarrollo de nuevas features hasta que la deuda técnica de cobertura alcance el 60%.
Curva de Aprendizaje (Herramientas) El equipo tarda en adoptar las nuevas herramientas (Postman/Cypress).	Alta	Medio	Cross-Training: Los perfiles Senior mentorizan a los Junior (Programa de Padrinos). Dedicar 10% del Sprint a capacitación.	Consultoría Externa: Contratación temporal de un experto (Staff Augmentation) para acelerar la configuración inicial del framework.

7.3 Matriz de decisiones ante riesgos materializados

Para agilizar la toma de decisiones durante el Sprint, establecemos la siguiente política:

- **Si un Riesgo Crítico (P1) se materializa 24hs antes del Release**: Decisión: NO GO. El lanzamiento se cancela automáticamente. No se requiere reunión de comité para decidirlo. La calidad prima sobre la fecha.
- **Si un Riesgo Alto (P2) se materializa:** Decisión: GO con Release Note. Se libera a producción, pero se agrega una nota en el "Known Issues" y se prioriza el Hotfix para la siguiente ventana.

8. Reporte de defectos

Todo defecto será documentado en la herramienta de gestión (Jira) siguiendo un estándar para facilitar su corrección.

8.1 Ciclo de vida del defecto

Definimos los estados por los que viaja un bug:

- New (Nuevo): El defecto es creado por QA o detectado en producción.
- Triage (Análisis): El Tech Lead o QA Lead revisa el ticket. Se decide si se arregla ahora, se pospone (Backlog) o se rechaza (Works as Designed).
- In Progress (En Progreso): El desarrollador está trabajando en el fix.
- Ready for QA (Listo para Pruebas): El desarrollador ha desplegado el fix en el entorno de Staging. El desarrollador NO cierra el ticket.
- Verified (Verificado): QA re-ejecuta las pruebas.
    - Si funciona: Se cierra el ticket.
    - Si falla: Se reabre (Reopened) y vuelve a prioridad alta.
- Closed (Cerrado): El ciclo ha terminado exitosamente.

8.2 Planilla obligatoria

- Campos Obligatorios: ID, Título (descriptivo y único), Severidad, Prioridad, Pasos para reproducir, Resultado Esperado vs. Obtenido, Evidencia (Logs/Screenshots) y Entorno (versión, dispositivos, SO, conexión, usuario de prueba).

8.3 Bug Triage

Para evitar que Jira se convierta en un cementerio de bugs olvidados cómo el sistema es actualmente con muchos bugs sin resolver se propone:

- Reunión semanal de triaje: QA Lead y Product Owner revisan los defectos nuevos P3 y P4. Se decide si se arreglan en el próximo Sprint o se mueven al Backlog profundo.
- Política de "Defectos Fantasma": Si un defecto no se puede reproducir después de 3 intentos por parte del desarrollador y 1 intento conjunto (Pair Debugging), se cierra como "Cannot Reproduce".

8.4 Metricas de calidad de defectos

QA también será evaluado por la calidad de sus reportes.

- **Tasa de Rechazo (Rejection Rate):** Porcentaje de bugs devueltos como "No es un bug" o "Falta información". (Objetivo: < 5%).
- **Bug Aging (Envejecimiento):** Tiempo promedio que un bug P1/P2 permanece abierto. (Objetivo P1: < 24hs).

9. Criticidad y urgencia de las pruebas

9.1 Nivel de criticidad

Definimos la gravedad del defecto basándonos en la pérdida de funcionalidad y dinero.

Nivel	Etiqueta	Definición operativa	Criterios de clasificación
C1	Crítica (blocker)	El negocio se detiene. El usuario no puede completar los flujos esenciales (core). No hay solución alternativa (workaround).	· falla en pagos/suscripción. · falla en login/registro. · video no reproduce (crash/pantalla negra). · vulnerabilidad de seguridad (datos expuestos).
C2	Alta (major)	Funcionalidad degradada severamente. Una característica importante falla, pero el núcleo del negocio sigue operando o existe un workaround complejo.	· falla en búsqueda o "mi lista". · subtítulos desincronizados. · error en dispositivos tier 1 (samsung/lg). · problemas de performance notables (lentitud).
C3	Media (minor)	Problema de usabilidad. La funcionalidad existe, pero es incómoda o confusa. Defectos visuales que no impiden el uso.	· traducciones incorrectas (localización). · errores visuales en la interfaz (UI). · fallos en dispositivos tier 2.
C4	Baja (trivial)	Cosmético. Defectos estéticos menores o problemas en escenarios muy poco probables.	· errores ortográficos menores. · desalineación de píxeles. · fallos en dispositivos tier 3 (legacy).

9.2 Niveles de urgencia de ejecución

Se define el orden en que el equipo de qa debe ejecutar los casos de prueba o verificar los fixes.

- **U1 - inmediata (hotfix/smoke):** se prueba ahora mismo. Detiene cualquier otra tarea. Se ejecuta en el momento que desarrollo entrega el build. Aplica a: smoke tests automáticos y verificación de defectos C1.
- **U2 - dentro del sprint:** debe probarse y cerrarse antes de que termine el sprint actual. No puede pasar al siguiente. Aplica a: nuevas historias de usuario comprometidas y defectos C2.
- **U3 - próximo release:** puede esperar a la siguiente ventana de lanzamiento programada. Aplica a: defectos C3 y deuda técnica menor.
- **U4 - backlog**: se probará si sobra tiempo o en fases de "hardening" (estabilización) futuras.

9.3 Matriz de decisión

	Urgencia Inmediata (U1)	**Urgencia Alta (U2)**	**Urgencia Baja (U3/U4)**
Criticidad Crítica (C1)	¡EMERGENCIA! QA detiene todo. Se prueba el Hotfix en Producción/Staging inmediatamente.	Alta Prioridad Debe quedar automatizado en este Sprint.	Planificar re-arquitectura urgente.
Criticidad Alta (C2)	Prioridad Alta Probar antes de cerrar el día.	Flujo Normal Probar dentro del ciclo del Sprint.	Deuda Técnica Agendar para el próximo Q.
Criticidad Media (C3)	Probar rápido (Quick fix).	Relleno Probar si sobra tiempo en el Sprint.	Backlog Se prueba en fases de "limpieza".
Criticidad Baja (C4)	No se prueba	No se prueba	Won't Fix Se cierra el ticket.

10.Ciclo de vida de desarrollo y pruebas

Adoptamos un enfoque de **testing continuo**. Las actividades de calidad comienzan en el momento en que se concibe una idea y no terminan hasta que el producto está en producción y monitoreado.

El ciclo de vida se ejecuta dentro de cada iteración (sprint de 2 semanas) y consta de 5 fases sincronizadas:

1. **Fase 1: análisis y definición (shift-left):** durante el refinamiento del backlog y planning QA actúa como consultor de calidad para prevenir defectos de diseño.
    - **Revisión de historias:** qa verifica que cada historia de usuario cumpla con el DoR. Si falta información sobre dispositivos o lógica de negocio, se rechaza.
    - **Diseño de escenarios (ATDD):**
        - QA redacta los **criterios de aceptación** en formato Gherkin (*dado/cuando/entonces*) junto con el product owner.
        - Se define la **matriz de cobertura**
    - **Salida:** escenarios de prueba aprobados antes de escribir una línea de código.
2. **Fase 2: diseño y preparación (test design):** paralelo al desarrollo mientras los desarrolladores codifican, QA prepara el terreno. No esperamos a recibir el binario.

    - **Automation engineers:**

    - Escriben los scripts de prueba de API (postman/restassured) basándose en el contrato Swagger definido en la fase 1.
    - Preparan los datos de prueba (mocks) para la integración continua.
  - **Manual/exploratory testers:**
    - Diseñan las sesiones exploratorias en smartTVs.
    - Configuran el laboratorio de dispositivos físicos.
3. **Fase 3: ejecución continua (test execution):** en cuanto hay un despliegue en testing código llega al entorno de QA se activa el protocolo de validación.
  - **Filtro automático (smoke test):**
    - Al hacer *merge*, el pipeline CI/CD ejecuta automáticamente los tests unitarios y la suite de API crítica.
      - *Si falla:* el despliegue se rechaza automáticamente. QA no interviene manualmente.
      - *Si pasa:* se notifica al equipo que la versión es estable para pruebas profundas.
  - **Ejecución manual (nuevas features):**
    - Se ejecutan las sesiones exploratorias en dispositivos **Tier 1** (Samsung/LG/IoS) enfocándose en la usabilidad y calidad de video.
  - **Gestión de defectos:**
    - Reporte inmediato en jira siguiendo la plantilla obligatoria.
    - Validación de fixes (re-test) con prioridad U1/U2
4. **Fase 4: regresión y estabilización (hardening):** Durante últimos días del sprint aseguramos que lo nuevo no rompió lo viejo.
  - **Regresión automática:** ejecución completa de la suite de regresión (API + UI web crítica).
  - **Sanidad en Tier 2:** realización de un smoke test rápido manual en dispositivos secundarios (ej. Tablets Android, TVs viejas) para asegurar que no hay bloqueos catastróficos.
  - **Code freeze:** no se acepta código nuevo, solo fixes de bugs críticos (C1).
5. **Fase 5: cierre y retrospectiva:** Al finalizar el sprint.
  - **Demo:** QA presenta los resultados de calidad y métricas de performance y UX.
  - **Mantenimiento de activos:**
    - Los scripts automatizados nuevos se integran a la rama main.
    - Se actualizan los datos de prueba para el siguiente sprint.
  - **Retrospectiva de calidad:** análisis de root cause de los bugs escapados.

10.1 Flujograma de responsabilidades (RACI)

Actividad	Dev Team	QA Automation	QA Manual	Product Owner
Unit Tests	R (Responsable)	C (Consultado)	I (Informado)	I
Definir Gherkin	C	C	R	A (Aprueba)
Scripting API	I	R	I	I
Pruebas en TV	I	I	R	C
Aprobar Release	I	C	R	A

R: Responsable: quien realiza la tarea. A: Accountable: quien aprueba la tarea. C: Consultado: quien da feedback. I: Informado.

Sección 2: Testing de Videojuegos e Inteligencia Artificial: Horizon Zero Dawn

1. Propósito y descripción

El objetivo de este informe es documentar la estrategia de validación integral para un videojuego de mundo abierto en tercera persona. El análisis se centra en dos pilares: la calidad general (experiencia sensorial y jugabilidad) y la validación profunda de los sistemas de inteligencia artificial de los enemigos (máquinas), elementos centrales de la propuesta de valor del juego.

2. Ficha técnica de las pruebas

2.1 Entorno de hardware (Test Bed)

- Consola: Sony PlayStation 4 (Modelo Original - Revisión C-Chassis). Modelo: CUH-1216G (Versión Europea). Software actualizado versión de sistema 13.02.
- Mando: DualShock 4 Wireless Controller v1 (Modelo: CUH-ZCT1E).
- Conectividad: 2.4GHz con 300 Mbps teóricos.
- Visualización: TV 50" (Panel LED/OLED según corresponda) con soporte HDR10.
- Resolución: 1920 x 1080p (FHD).
- Software: Horizon Zero Dawn Limited Edition - PlayStation 4

3. Alcance del Testing

El alcance se divide en dos áreas de investigación:

1. **Testing General:** Audio, Controles, Gráficos y Localización.
2. **Testing de IA:** Comportamiento de enemigos, navegación (Pathfinding) y máquinas de estados.

4. Sesión de testing exploratorio 1: Hoja de sesión

Duración: 60 min

Tester: Rodriguez Natacha

Fecha: 03/12/2024

4.1 Misión

Objetivo: Explorar el comportamiento de la IA de los "Vigías" (Watchers) y la respuesta de los controles durante el combate cuerpo a cuerpo y a distancia, verificando la sincronización de audio y colisiones.

4.2 Configuración del Entorno (Setup)

Condiciones iniciales, vital para reproducir fallos de hardware o disco.

Consola: PS4 Standard (Ficha técnica especificada anteriormente)

Medio: Disco Físico insertado.

Instalación: Juego base instalado + Parche del día 1.

Idioma Consola: inglés.

Control: DualShock 4 (Batería al 100%, conectado vía Bluetooth).

4.3 Registro de Pruebas (Test Log)

Pruebas realizadas:

- Pruebas de Carga e Integridad del disco:
  - Inicié el juego desde el Dashboard de PS4. Verifiqué tiempo de carga inicial.
  - Observación: La cinemática de introducción se reprodujo sin "saltos de sonido" ni distorsión, confirmando buena lectura del Blu-ray.
- Pruebas de controles y ergonomía:
  - Probé la sensibilidad de los sticks analógicos (L3/R3) para mover a Aloy y la cámara.
  - Verifiqué el mapeo de botones: "X" para saltar, "Cuadrado" para agacharse (sigilo).
- Prueba de Estrés: Realicé una secuencia rápida: Rodar -> Apuntar -> Disparar.
  - Resultado: Respuesta fluida, sin input lag perceptible. La vibración del DualShock respondió al tensar el arco (Prueba de Haptics).
- Pruebas de IA y Mecánicas de Combate (Gameplay):
  - Me acerqué sigilosamente a un Vigía. Lancé una piedra para probar su sistema de "Alerta" (cambio de luz azul a amarilla).
  - Observación IA: El enemigo investigó el punto de impacto del sonido y luego volvió a su ruta correctamente.
- Ataqué cuerpo a cuerpo (R1).
  - Verifiqué la Hitbox. El golpe conectó visualmente y la barra de vida del enemigo bajó (Coherencia visual/lógica).
  - Verifiqué que los sonidos de impacto (metal contra metal) se reprodujeron en sincronía con la animación (Sincronización labial/efectos).
- Pruebas de Gráficos y Entorno:
  - Forcé colisiones: Corrí hacia árboles y rocas para ver si Aloy los atravesaba (Clipping).
  - Evalué la iluminación: Entré a una cueva oscura para verificar si la iluminación dinámica cambiaba correctamente.
- Pruebas de Localización (L10N):
  - Activé los subtítulos en español.
  - Verifiqué que el texto de los diálogos coincidiera con el audio (Doblaje) y que no hubiera desbordamiento de texto en la interfaz.

4.4 Defectos encontrados:

No se encontraron defectos bloqueantes ni visuales en el alcance de esta sesión

4.5 Notas de calidad y feedback

- Feedback de Audio: Aunque no hubo errores, noté que la mezcla de sonido es excelente; los pasos de las máquinas se escuchan claramente a la distancia, lo que facilita la preparación del combate.
- Feedback Visual: La transición de iluminación al entrar a cuevas es suave y realista, no afecta la visibilidad del jugador.
- Usabilidad (UX): El mapeo del botón "Círculo" para interactuar se siente cómodo y consistente con otros juegos del género.

5. Sesión de testing exploratorio 2: Hoja de sesión

Duración: 60 min

Tester: Rodriguez Natacha

Fecha: 09/12/2024

5.1 Misión

Objetivo: Validar la respuesta de los controles al escalar (zonas amarillas) y usar el arco, asegurar que la recolección de recursos actualice el inventario correctamente y confirmar que los objetivos de la misión progresen sin bloqueos.

5.2 Configuración del Entorno (Setup)

Condiciones iniciales, vital para reproducir fallos de hardware o disco.

Consola: PS4 Standard (Ficha técnica especificada anteriormente)

Medio: Disco Físico insertado.

Instalación: Juego base instalado + Parche del día 1.

Idioma Consola: inglés.

Control: DualShock 4 (Batería al 100%, conectado vía Bluetooth).

5.3 Registro de Pruebas (Test Log)

- Pruebas de navegación y entorno (Parkour):
    - Buscar salientes marcadas con pintura amarilla/cuerdas y presionar "X" para saltar hacia ellas. Se evalúan las colisiones y la animación. Aloy se agarra automáticamente a la superficie amarilla. Sus manos coinciden visualmente con el borde (sin "flotar" o atravesar la textura).
    - Mover el stick analógico hacia otro borde amarillo y saltar. Observación: La transición de animación entre agarres es fluida. No hay "clipping" (atravesar objetos) con la roca.
- Pruebas de combate a distancia (Arco y Piedras):
    - Apuntar con el arco (L2) y tensar la cuerda al máximo (R2). Pruebas de Controles (Haptics/Vibración). Validación: El control DualShock 4 vibra

incrementando la intensidad a medida que se tensa el arco. La retícula se cierra para indicar precisión máxima.
- Seleccionar "Piedra" en la rueda de herramientas y lanzarla (R2) cerca de una máquina. Mecánica de distracción y física. Observación: La trayectoria visual (línea blanca) coincide con el lugar donde cae la piedra. Se reproduce el sonido de impacto correcto.

- Pruebas de Economía y Meta-Juego (Recolección):
  - Acercarse a una máquina caída o un arbusto de madera y mantener "Triángulo" para recolectar Esquirlas de Metal y Madera. Mecánica Meta (Inventario) y Feedback de Interfaz.
  - Validación:
    1. Se reproduce la animación de "agarrar". **(NO Pass)**
    2. Aparece el icono del ítem a la derecha de la pantalla (UI). (PASS)
    3. Verificación Crítica: Abrir el menú y confirmar que el contador de esquirlas y madera subió exactamente la cantidad recogida. (PASS)
- Pruebas de Lógica de Misión (Objetivos)
  - Realizar la tarea activa (ej: "Reúne madera de risco"). Pruebas de progresión. Observación: Al obtener la cantidad requerida, el texto del objetivo en pantalla cambia de "0/3" a "3/3" inmediatamente y aparece el check de "Completado". Validación: El juego actualiza el siguiente paso de la misión (ej: "Fabrica flechas") sin necesidad de recargar. (Pass)

5.4 Reporte de defecto

ID: BUG-GP-001

Título: No es posible interactuar con máquina derrotada (Saquear) a pesar de tener el icono de botín activo.

Categoría: Gameplay / Interacción.

Severidad: Media

Frecuencia: 100%, siempre que sea la misma máquina.

Descripción: Al derrotar a una máquina un Galopador, aparece el icono de "Recursos Disponibles" (Rombo blanco con caja) sobre el cadáver. Sin embargo, al acercarse al cuerpo desde cualquier ángulo, no aparece el prompt del botón "Triángulo" en la interfaz, haciendo imposible recolectar el botín.

Precondiciones:

- Derrotar a una máquina que contenga botín.

Pasos para reproducir:

- Eliminar a la máquina.
- Identificar el icono de botín sobre el cadáver.
- Acercarse al cuerpo (colisionando con él).
- Buscar el ángulo para que aparezca la opción "Saquear".

Resultado Actual: El icono de botín persiste, pero el juego no reconoce la proximidad del jugador para habilitar la interacción. El jugador no puede obtener la recompensa.

Resultado Esperado: Al entrar en el radio de interacción del cadáver, debería aparecer el texto "Saquear" junto con el botón "Triángulo", permitiendo la recolección.

Evidencia: Video de la evidencia

5.5 Notas de calidad y feedback

La sesión de pruebas enfocada en las mecánicas Core (navegación, combate y economía) arroja un resultado general positivo en términos de estabilidad funcional, aunque se ha detectado un defecto de pulido visual que afecta la inmersión.

Resumen de Resultados:

- **Navegación y Entorno (PASS):** El sistema de parkour es robusto. Las pruebas de colisión y animación en zonas de escalada (pintura amarilla) demostraron una integración correcta entre el modelo del personaje y la geometría del nivel, sin presencia de *clipping* ni errores de física.
- **Combate a Distancia (PASS):** La respuesta de los controles es óptima. La integración del hardware (vibración háptica del DualShock 4) y las físicas de los proyectiles (trayectoria de piedras y flechas) funcionan según lo diseñado, proporcionando el feedback esperado al jugador.
- **Lógica de Misión (PASS):** El *scripting* de las misiones es sólido. La actualización de objetivos en la UI y la transición entre estados de la misión ocurren en tiempo real sin bloqueos ni necesidad de recargar.

Área de Atención (Defecto Detectado):

- **Economía y Recolección (PARCIAL):** Si bien la lógica backend funciona correctamente, los contadores de inventario se actualizan y el feedback de UI aparece, se ha identificado un fallo visual constante:
    - **Fallo:** La animación de "agarrar" no se reproduce al recolectar recursos en movimiento (**No Pass**).

- **Impacto:** Afecta la experiencia de usuario (UX), aunque no bloquea el progreso ni la economía del juego.

Se aprueba el estado funcional de las mecánicas de combate y navegación. Se deriva al equipo de Animación/Gameplay el reporte del defecto visual en la recolección para su corrección en el próximo ciclo de pulido.

6. Análisis y testing de inteligencia artificial (IA)

6.1 Investigación: ¿Cómo funciona la IA en HZD?

A diferencia de otros juegos donde los enemigos simplemente "patrullan y atacan", en Horizon la IA está construida bajo un sistema híbrido que combina HTN (Hierarchical Task Network) con decisiones basadas en utilidad. Esto es lo que necesitas saber para probarla:

6.1..1 Planificación HTN (red de tareas jerárquica)

En lugar de una simple "Máquina de Estados", Si te veo -> Ataco, las máquinas tienen "Objetivos" complejos que descomponen en tareas más pequeñas. Si un Recolector necesita huir, no solo corre en línea recta. Su "plan" incluye: Buscar ruta segura -> Evitar obstáculos dinámicos -> Mantenerse cerca de la manada -> Alertar a otros. Si se corta su ruta con trampas, el sistema del enemigo recalcula un nuevo plan como por ejemplo saltar la soga de la aturdidora.

6.1..2 Navegación dinámica

El juego utiliza un sistema de navegación que se adapta a agentes de distintos tamaños como un Vigía o un Atronador en un terreno que cambia.

Las máquinas grandes destruyen árboles y rocas. La IA debe actualizar el mapa de navegación en tiempo real para saber que ahora puede pasar por donde antes había un árbol.

6.1..3 El sistema de Manada (Herd Mentality)

Lo más distintivo de HZD es que las máquinas no actúan solas; forman un ecosistema. Investigando la arquitectura, encontramos que se dividen en roles específicos:

1. Adquisición Pastadores, Galopadores. Son dóciles. Su prioridad es huir.
2. Reconocimiento: Vigías, Cuernilargos. Su función es detectar amenazas y alertar al grupo.
3. Combate: Dientes Serrados, Atronadores. Responden a las alertas defendiendo a la manada.

Las máquinas pueden solicitar unirse a manadas existentes. Esto significa que, si matas a los defensores, una máquina de combate cercana podría "adoptar" al grupo de pastadores indefensos.

6.2 Estrategia de pruebas de IA

En función a la investigación realizada sobre la Ia en los enemigos se proponen las siguientes pruebas

6.2..1 Pruebas de autonomia y límites operativos

Se propone la prueba de Leashing, correa invisible: evaluar la autonomía de la máquina al perseguir al jugador lejos de su zona de hábitat. Provocar a un Dientes Serrados por ejemplo y correr una gran distancia sin ocultarse. Se observaría un fallo si la máquina persiga infinitamente por todo el mapa, rompiendo su límite operativo, o que se quede "congelada" al llegar al límite invisible.

6.2..2 Prueba de adaptabilidad del entorno

Se propone la prueba de pathfinding dinámico. Ver si la IA recalcula su ruta ante cambios físicos. Hacer que una máquina persiga al personaje y destruir un obstáculo destructible (un árbol o muro) en su camino, o creando algún bloqueo. Evaluar cuanto tiempo tarda el sistema en adaptarse a este cambio. Se observaría un fallo si la máquina intenta caminar chocando contra el aire donde antes había un árbol, o que ignore el nuevo obstáculo.

6.2..3 Prueba de sistema no determinísticos

Realizando una prueba de consistencia de comportamiento. Se busca verificar la variabilidad aceptable en los ataques. Una prueba podría ser cargar la partida varias veces, ejemplo 5. En cada una, pararse exactamente en el mismo punto frente a un Vigía y dejar que te vea. Falla si 4 veces ataca y la quinta vez se queda quieto o huye sin razón lo que los resultados están fuera de los límites aceptables.

6.2..4 Prueba de ataques adversariales

Se propone la prueba de confusión de sensores (stealth spam). Se busca enloquecer el sistema de detección. Por ejemplo, entrar y salir del campo de visión de un enemigo (agacharse/pararse) a una velocidad muy alta (spam de botones) en el borde de la hierba alta. Falla si por ejemplo, el indicador de alerta parpadee erráticamente, que la IA cancele su ataque a mitad de animación, o que ignore al jugador por completo.

6.2..5 Prueba de integración de componentes

La prueba de reacción a estímulos, se busca validar que el "input" de sonido llegue al modelo de decisión. Disparar una flecha a una superficie metálica detrás de un enemigo ciego por lo que el componente de radar esta destruido. Falla si el enemigo no reaccione al sonido fuerte, indicando una falla en la integración entre el motor de audio y la lógica de la IA.