Fase de Análisis de Requerimiento

Cliente	Oregon Trail Survival
Usuario	Jugador
Requerimientos funcionales	RF1. Movimiento en cuatro direcciones RF2: Gestión de escenarios. RF3: Sistema de puntería RF4. Gestión de munición escasa (preguntar si se pueden los requerimientos 2 y 3) RF5. Recarga del arma RF6. Sistema de vidas RF7. Gestión de inventario (priorizar recursos limitados) RF8. Sistema de recolección RF9. Generación aleatoria de enemigos RF10. Comportamiento de los enemigos RF11. Sistema de logros
Contexto del problema	Oregon Trail Survival es una adaptación del clásico juego educativo The Oregon Trail que incorpora elementos de juegos de acción y supervivencia en tiempo real. Esta versión fusiona la gestión de recursos y la toma de decisiones estratégicas del juego original con mecánicas de combate y exploración en un entorno de múltiples escenarios interconectados.
Requerimientos no funcionales	● El sistema debe tener una interfaz amigable para la facilidad de su uso ● El sistema debe garantizar un tiempo de respuesta menor a 1 segundo en acciones críticas. ● El sistema debe ejecutarse en Java utilizando JavaFX para la interfaz gráfica. ● El sistema debe implementar hilos y concurrencia para animaciones y ejecución fluida. ● El sistema debe desarrollarse bajo la metodología TDD, con pruebas unitarias en JUnit. ● El sistema debe mostrar mensajes claros y entendibles para el jugador en eventos críticos como muerte, victoria y falta de recursos. ● El sistema debe ser mantenible, aplicando buenas prácticas y patrones de diseño vistos en el curso. ● El sistema debe incluir un manual de usuario con controles, indicadores y estrategias de juego accesible desde el menú. ● El sistema debe integrar el API de Gemini para generar diálogos entre los enemigos del juego.
Requerimientos de proceso	● Reporte de indicadores de calidad en 15 commits, documentados en el README. ● Entregas parciales: ○ Semana 10: Requerimientos, pruebas y diagrama de clase. ○ Semana 14: Actualización del diseño, estructuras (listas, árboles), ordenamientos, búsqueda binaria y lógica implementadas. ○ Semana 18: Entrega final y sustentación del desarrollo del juego. ● La evidencia de los commit tiene que ser equitativa entre todos los integrantes del proyecto, debe ser trabajado mediante la herramienta de Git y subido a un proyecto en GitHub.

Cliente	Oregon Trail Survival
	• En caso de uso de IA generativa, llevar registro de los resultados arrojados, incluyendo prompts proporcionados.

RF1

Identificador y nombre	RF1. Movimiento en cuatro direcciones		
Resumen	*El juego debe permitir el movimiento en cuatro direcciones o en 2 dimensiones (arriba, abajo, izquierda y derecha) para el desplazamiento del jugador por los distintos escenarios*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	mov_arriba	char (Tecla asignada)	W/flecha arriba
	mov_abajo	char (Tecla asignada)	S/flecha abajo
	mov_izquierda	char (Tecla asignada)	A/flecha izquierda
	mov_derecha	char (Tecla asignada)	D/flecha derecha
Resultado o Postcondición	Movimiento del usuario mediante		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	ubicacion_Jugador	int[][] (matriz de x*y)	*Nueva ubicación en la matriz (escenario)*

RF2

Identificador y nombre	*RF2*-gestión de munición escasa		
Resumen	*El sistema debe permitir al jugador gestionar de manera eficiente la munición disponible, mostrando en todo momento la cantidad restante. La munición es limitada y debe tratarse como un recurso escaso que impacta en la estrategia de juego. Si la munición llega a cero, el jugador no podrá disparar hasta recolectar más o cambiar de arma, garantizando una supervivencia realista.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	cantidad_MunicionActual	Entero(Int)	*Número ≥ 0*
	cantidad_MunicionMaxima	Entero(Int)	*Número > 0*
	tipo_arma	String	“rifle” o “revólver”

	accion_Jugador	String	“disparar”, “recargar”, “recolectar”
Resultado o Postcondición	El sistema debe actualizar correctamente el inventario de municion despues de cada accion como disparo, recarga o recolección. En caso de quedarse sin municion, se mostrara un mensaje de advertencia al jugador indicando que no puede disparar.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	mensaje_municion	String	*Te quedan x disparos disponibles, o sin municion, recargue o busque mas.*
	municion_actualizada	Entero (Int)	Número que representa la munición restante después de la acción.

RF3

Identificador y nombre	RF3. Sistema de puntería		
Resumen	*El juego debe implementar un sistema de puntería en el que la retícula del arma siga en tiempo real el movimiento del mouse, permitiendo disparar en la dirección en que se encuentre el objetivo.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	movimiento_mouse	int[]	Movimiento en tiempo real del mouse dentro del área de juego
	punteria	char(tecla asignada)	clic derecho para disparar
Resultado o Postcondición	El juego debe ejecutar un disparo en la dirección indicada por la coordenada, disminuyendo la munición disponible e impactando al enemigo si está en el rango.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	mensaje_disparo	String (String)	*"Disparo realizado con éxito" o "Sin munición disponible"*
	efecto_visual	Animación	Aparición de disparo en pantalla con trayectoria

RF4

Identificador y nombre	RF4. Gestión de munición escasa		
Resumen	*El juego debe dejar evidenciado e implementado todo sobre la gestión de la munición, tal como un recurso limitado y que se encuentra escaso dentro de todos los escenarios del juego.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	disparar	char (Tecla asignada)	click izquierdo
	recoger_municion	char (Tecla asignada)	E
Resultado o Postcondición	El jugador realiza la acción de disparo o de recogida, cambian los valores de inventario (munición gastada o munición recogida), si la munición es igual a “0”, se debe bloquear el disparo y no cambian los valores. El disparo se efectúa o se bloquea dependiendo del estado de la munición del arma, en tal caso de bloquearse, tira un mensaje en pantalla.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	disparo_fallido	String (String)	*“Sin munición”*

RF5

Identificador y nombre	RF5. Recarga del arma		
Resumen	*El juego debe contar con una mecánica de recarga, ya sea automática o manual dependiendo de la elección del jugador con la ayuda de la configuración del juego, está tomará como parámetro una tecla asignada para la recarga..*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	recarga	char (Tecla asignada)	R
Resultado o Postcondición	El arma debe efectuar una recarga dependiendo de las balas disponibles en inventario, si las balas disponibles es igual a 0, el arma no será recargada, en cambio, si las balas disponibles es mayor a 0, entonces se podrá recargar sin ningún problema, sin embargo, si el arma cuenta con munición llena, lanzará un mensaje.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	recarga_fallida	String (String)	*“No cuenta con suficiente munición para recargar”*
	recarga_llena	String (String)	*“Arma cargada”*

RF6

Identificador y nombre	RF6. Sistema de vidas		
Resumen	*El juego debe contar con un sistema de vida, el jugador contará con 3 impactos antes de morir, cuando sus 3 vidas se acaben, debe lanzar un mensaje de juego terminado.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	vidas	Int (Integer)	vidas = 3
Resultado o Postcondición	Si el jugador pierde una vida, el juego debe notificar al jugador por medio de un mensaje, y cuando ese contador de vidas llegue a 0, debe lanzar una pantalla de juego terminado		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	vida_perdida	String (String)	*“Ten más cuidado, has perdido una vida, vidas restantes: ”+vidas=vidas-1+*
	vida_perdida	Int (vida < 1)	*“Has perdido, inténtalo nuevamente”*

RF7

Identificador y nombre	RF7. Gestión de inventario		
Resumen	*el juego debe tener la capacidad de ordenar o gestionar el inventario del jugador por orden de prioridad con respecto a los recursos u objetos los cuales tenga en su inventario*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	prioridad_de__cual_recurso	int	un número de 1 a 3, donde el 1 es medicamentos, 2 medicinas y 3 municion
Resultado o Postcondición	Mostrar el inventario organizado en base a la prioridad asignada, y aumentar la cantidad de recursos, para así no permitirle al usuario, recolectar recursos no prioritarios si no hay espacio .		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	mensaje_Confirmacion_de_ordenamiento	String (String)	*El inventario se organizó en base al recurso prioritario + prioridad_de_cual_recurso+ “...”*

RF8

Identificador y nombre	RF8. Sistema de recolección		
Resumen	*El juego debe permitirle al usuario poder recolectar los recursos disponibles en los escenario o en los mapas, en caso tal que tenga el espacio suficiente para poder meterlos o recolectarlos en su inventario*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	recurso_recolectado	Recurso(recurso)	que sea un recurso y sea posible recolectarlo
Resultado o Postcondición	si el recurso se puede recolectar por que aun hay capacidad en el inventario, se debe mostrar el inventario y por ende el nuevo recurso ahi guardado; sino es por que el espacio del inventario esta lleno		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	mensaje_De_inventario_Lleno	String (String)	*El inventario está lleno, tire algun recurso para poder recoger el objeto “ + recurso_recolectado+”*
	mensaje_de_exito	String(String)	*se pudo guardar el recurso “ +recurso_recolectado + ”*

RF9

Identificador y nombre	RF9. Generación aleatoria de enemigos		
Resumen	*El sistema debe generar enemigos de manera aleatoria en cada partida, garantizando variabilidad en la experiencia. Los enemigos aparecerán en diferentes posiciones de los escenarios y tendrán un comportamiento básico de persecución y ataque al jugador.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	escenario_actual	String (Enum)	1. “Llanuras”, 2.“Montañas Rocosas”, 3. “Río Columbia”
	autor	String (String)	texto
	ISBN	String (String)	texto
	materia	String (String)	texto

	número_ Copias_Disponibles	Entero(Int)	Un número >= 0
Resultado o Postcondición	El sistema genera y coloca en el escenario una cantidad de enemigos aleatoria dentro de los límites establecidos, listos para interactuar con el jugador.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	lista_enemigos	Lista (objetos)	{enemigoID: 1, tipo: “autómata”, posición: (x,y)
	mensaje_enemigos	String	“Se generaron 5 enemigos en llanuras**”**
	animacion_spawn	animación	Enemigos apareciendo en pantalla con efecto visual

RF10

Identificador y nombre	RF10. Comportamiento de los enemigos		
Resumen	*el juego debe hacer que los enemigos persigan al jugador y que estos le hagan daño dependiendo del arma, o del sistema definido del daño al jugador, el enemigo no se puede sobreponer en la misma ubicación al jugador*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	Ubicación_jugador	int[][]	una posición que este dentro de la matriz o del escenario
Resultado o Postcondición	Mostrar visualmente o en texto si el enemigo alcanzó o atacarte y hacerte daño, sino solo se muestra visualmente cómo el enemigo te persigue		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	mensaje_de_daño	String (String)	*un enemigo te ha hecho “..” daño*
	nueva_ubicacion_Enemigo	int[][]	*(se ve visualmente la nueva ubicación)*

RF11

Identificador y nombre	RF11. Sistema de logros		
Resumen	*El sistema debe registrar y mostrar los logros obtenidos por el jugador, de acuerdo con un criterio definido como el : puntaje, enemigos eliminados, recursos recolectados. Los logros deben visualizarse en una ventana independiente.*		
Entradas	**Nombre entrada**	**Tipo de dato**	**Condición valores válidos**
	accion_jugador	String	“enemigo eliminado”, “nivel superado”, “recurso recolectado”
	puntaje_actual	Entero (int)	Puntaje acumulado del jugador
Resultado o Postcondición	El sistema debe evaluar si la acción del jugador cumple las condiciones para obtener un logro. En caso afirmativo, se inserta un nuevo nodo en el árbol binario de logros y se actualiza la ventana de logros.		
Salidas	**Nombre salida**	**Tipo de dato**	**Formato**
	lista_logros	String	logroID: 1, nombre: “Cazador”, criterio: “5 enemigos eliminados”
	mensaje_logro	String	“Nuevo logro desbloqueado”

Método Dorfman

Requerimientos

<u>Movimiento y exploración:</u>

RQ1: El juego debe permitir el movimiento en cuatro direcciones (arriba, izquierda, derecha y abajo).

RQ2: El juego debe contar con tres escenarios:

- Escenario 1: Llanuras y praderas

- Escenario 2: Montañas rocosas
- Escenario 3: Río Columbia y áreas cercanas a Oregón

Siendo en este mismo orden, la etapa inicial del viaje, parte media del viaje y la etapa final del juego.

RQ3: El juego debe contar con transición entre etapas, esta será mediante puertas de acceso, pasajes montañosos o vados fluviales.

Sistema de Combate y armas:

RQ4: El juego debe contar con dos tipos de armas:

- Rifle de Avancarga: Arma que cuenta con un mayor daño, pero menor velocidad de disparo.
- Revolver: Arma que cuenta con un menor daño, pero con una mayor velocidad de disparo.

RQ5: El juego debe permitir tener un sistema de puntería en base a la retícula del mouse.

RQ6: El juego debe dejar evidenciado e implementado sobre la gestión de la munición como un recurso limitado y que se encuentra escaso, además, después de haberse acabado la munición, debe contar con una mecánica de recarga, ya sea automática o manual dependiendo de la elección del jugador en la configuración del juego.

Supervivencia y Gestión de Recursos

RQ7: El juego debe contar con un sistema de vida, 3 impactos antes de morir

RQ8: El juego debe contar con un inventario el cual debe priorizar suministros esenciales como la comida, munición o curas.

RQ9: El juego debe poder recolectar recursos del entorno como comida, munición o curas.

RQ10: El juego debe usar la munición como un recurso escaso; es decir cada disparo debe reducir la cantidad de munición.

Autómatas Enemigos:

RQ7: El juego debe generar aleatoriamente enemigos en cada partida.

RQ8: Los enemigos perseguirán y atacarán al jugador, generando un daño al jugador.

Condiciones del juego:

RQ9: El juego debe mostrar una pantalla inicial con las opciones para iniciar juego, ver árbol de logros o salir.

RQ10: El juego debe mostrar los escenarios del juego con indicadores de vida, el inventario y las municiones.

RQ11: El juego debe mostrar una pantalla de derrota cuando el jugador pierda toda la vida o cumpla todas las condiciones para que sea derrotado.

RQ12: El juego debe mostrar una pantalla de victoria cuando el jugador cumpla los objetivos definidos.

RQ13: El juego debe mostrar el árbol de logros del jugador en una ventana independiente usando árboles binarios de búsqueda.

RQ14: El juego debe almacenar y organizar los logros en árboles binarios de búsqueda.

RQ15: El juego debe integrar diálogos con los enemigos y los bots.

RQ16: El juego debe ejecutar las animaciones.

RQ17: EL juego debe mostrar un manual de usuario.

ENTIDADES:

- Jugador (movimiento, indicadores de vida)
- Escenarios
- Armas
- Recurso.
- Enemigos
- Árbol de logros.

SISTEMA: THE OREGON TRAIL

SUBSISTEMAS:

- S1: Sistema de movimiento
- S2: Sistema de escenarios
- S3: Sistema de combate y armas
- S4: Sistema de enemigo
- S5: Sistema de recursos
- S6: Sistemas de logros

Requerimientos	S1	S2	S3	S4	S5	S6
RQ1	x	x		x		
RQ2	x	x		x	x	
RQ3	x	x			x	
RQ4			x	x	x	
RQ5	x		x	x		
RQ6			x		x	
RQ7				x	x	
RQ8				x		
RQ9		x				x
RQ10		x			x	
RQ11		x				
RQ12		X				X
RQ13						X

RQ14						X
RQ15		X		X		
RQ16	X	X	X			
RQ17		X				X

Subespecificación de requerimientos (Programas)

RQ1 — Movimiento en 4 direcciones y colisiones

• RQA_1: El jugador puede moverse arriba si la celda/espacio destino es transitable.

• RQB_1: El jugador puede moverse abajo si la celda/espacio destino es transitable.

• RQC_1: El jugador puede moverse a la izquierda si la celda/espacio destino es transitable.

• RQD_1: El jugador puede moverse a la derecha si la celda/espacio destino es transitable.

• RQE_1: El estado del juego mantiene la jugabilidad (sin soft-locks al moverse).

• RQF_1: El sistema impide atravesar paredes o salir de los límites del mapa.

• RQG_1: El mapa carga paredes/límites en su representación de colisión.

RQ2 — Tres escenarios y orden de avance

• RQA_2: Existen Llanuras, Montañas Rocosas y Río Columbia/áreas cercanas a Oregón.

• RQB_2: Hay puntos de acceso/portales para cambiar de escenario.

• RQC_2: El avance se impone en el orden histórico (inicio → medio → final).

RQ3 — Transiciones entre escenarios

• RQA_3: Al colisionar con un punto de acceso válido, se transfiere al siguiente escenario.

• RQB_3: Se persiste el estado del jugador (salud, inventario, munición) al transicionar.

• RQC_3: Se bloquean transiciones no permitidas (p. ej., saltar del inicio al final).

RQ4 — Dos tipos de armas

• RQA_4: Rifle: mayor daño base, menor cadencia.

• RQB_4: Revólver: menor daño base, mayor cadencia.

• RQC_4: Conmutación de arma activa (rifle ↔ revólver).

• RQD_4: Las estadísticas de armas (daño, cadencia, capacidad/recarga) son parametrizables.

RQ5 — Puntería por retícula del mouse

• RQA_5: Mostrar retícula visible en el centro al inicio.

• RQB_5: La retícula sigue la posición del mouse en tiempo real.

• RQC_5: El arma del jugador se orienta hacia la retícula para disparar.

RQ6 — Munición escasa + recarga

• RQA_6: La munición es un recurso limitado por tipo de arma.

• RQB_6: Cada disparo reduce la munición disponible.

• RQC_6: Recarga manual (input del jugador) y opcional recarga automática (configuración).

• RQD_6: La recarga bloquea el disparo hasta completar el tiempo de recarga.

• RQE_6: Si no hay munición, el intento de disparo falla y se notifica en UI/HUD.

RQ7 (Salud) — 3 impactos antes de morir

• RQA_7(Salud): El jugador inicia con 3 puntos de vida.

• RQB_7(Salud): Al recibir daño, la vida decrementa en 1.

• RQC_7(Salud): Con 0 vida, se dispara estado Derrota (ver RQ11).

RQ8 (Inv) — Inventario con prioridad a esenciales

• RQA_8(Inv): El inventario tiene capacidad limitada (n° de slots o peso).

• RQB_8(Inv): Ítems esenciales: comida, munición, medicinas.

• RQC_8(Inv): Usar (consumir medicina/comida), equipar (munición/arma), descartar.

• RQD_8(Inv): Reglas de prioridad: cuando el inventario está lleno, se solicita gestión (p. ej., descartar).

RQ9 (Recursos) — Recolección de recursos del entorno

• RQA_9(Recursos): Spawns de loot (comida, munición, medicinas) por escenario.

• RQB_9(Recursos): Interacción (pickup) si hay capacidad.

• RQC_9(Recursos): Reglas de respawn/dispersión configurables para mantener la jugabilidad.

RQ10 — Munición como recurso escaso (cada disparo resta)

• RQA_10: Vincular la reducción de munición al evento de disparo.

• RQB_10: Evitar disparo si la munición es 0.

RQ7 (Enemigos) — Generación aleatoria "jugable"

• RQA_7(Enemigos): Spawner con densidad máxima por tiempo/área.

• RQB_7(Enemigos): Zonas seguras alrededor del jugador para evitar spawns injustos.

• RQC_7(Enemigos): Semilla configurable para reproducibilidad (si se requiere).

RQ8 (Enemigos) — Perseguir y atacar

• RQA_8(Enemigos): Comportamiento Perseguir: moverse hacia el jugador cuando está en rango.

• RQB_8(Enemigos): Atacar al estar a distancia efectiva.

• RQC_8(Enemigos): Daño por ataque definido (parámetro de balance).

RQ9 (UiMenu) — Pantalla inicial

• RQA_9(UiMenu): Iniciar juego, Árbol de logros, Salir.

RQ10 (HUD) — Indicadores en juego

• RQA_10(HUD): Mostrar vida, inventario (ítems clave) y munición.

RQ11 — Pantalla de derrota

• RQA_11: Mostrar derrota cuando vida llegue a 0 o se cumplan condiciones de fallo.

RQ12 — Pantalla de victoria

• RQA_12: Mostrar victoria cuando se cumplan objetivos definidos.

RQ13/14 — Árbol de logros (ABB)

• RQA_13: Visualizar árbol de logros en ventana independiente.

• RQB_14: Almacenar/organizar logros en árbol binario de búsqueda (insert/search/in-order).

RQ15 — Diálogos

• RQA_15: Sistema de diálogo activable con NPC/enemigos/bots.

RQ16 — Animaciones

• RQA_16: Reproducir animaciones de jugador, enemigos, UI transiciones.

RQ17 — Manual de usuario

• RQA_17: Mostrar manual accesible desde menú/pantalla de pausa.

Partición en módulos por subsistema

S1 — Sistema de movimiento

• S1.M1 Input: lectura de teclas/mouse (mover, usar, disparar, recargar).

• S1.M2 MovementController: aplica velocidad/dirección.

• S1.M3 Collision: consulta mapa y valida desplazamientos.

S2 — Sistema de escenarios.

• S2.M1 MapLoader: carga/representación (paredes, límites, spawn points).

• S2.M2 Gateways: puntos de acceso y reglas de transición.

• S2.M3 SceneState: persistencia de estado (salud, inventario, munición) al cambiar.

S3 — Sistema de combate y armas

• S3.M1 AimReticle: retícula y orientación del arma.

• S3.M2 FireControl: disparo/cadencia, evento de impacto.

• S3.M3 AmmoManager: conteos por arma, decrementos, validación de 0.

• S3.M4 Reloading: lógica y tiempos de recarga (manual/auto).

• S3.M5 DamageCalc: aplica daño a objetivo (jugador ↔ enemigo).

S4 — Sistema de enemigo

• S4.M1 Spawner: generación aleatoria con límites/semilla.

• S4.M2 EnemyAI: estados perseguir/idle.

• S4.M3 EnemyAttack: alcance, cadencia y daño al jugador.

S5 — Sistema de recursos

• S5.M1 Inventory: slots/capacidad, usar/descartar/equipar.

• S5.M2 LootPickup: interacción y validación de capacidad.

• S5.M3 LootSpawn: distribución/respawn de recursos por escenario.

S6 — Sistema de logros

• S6.M1 AchievementsBST: insertar/buscar/recorrer (ABB).

• S6.M2 AchievementsView: visualización en ventana independiente.

Segunda asignación (matriz sub-reqs ↔ módulos)

Sub-Req	Módulo dueño primario	Módulos de soporte
RQA_1 / RQB_1 / RQC_1 / RQD_1	S1.M2 MovementController	S1.M1 Input; S1.M3 Collision; S2.M1 MapLoader
RQE_1	S1.M2 MovementController	—
RQF_1 / RQG_1	S1.M3 Collision	S2.M1 MapLoader
RQA_2	S2.M1 MapLoader	S2.M3 SceneState
RQB_2	S2.M2 Gateways	S2.M3 SceneState
RQC_2	S2.M2 Gateways	S2.M3 SceneState

RQA_3	S2.M2 Gateways	S1.M2 MovementController
RQB_3	S2.M3 SceneState	S5.M1 Inventory; S3.M3 AmmoManager
RQC_3	S2.M2 Gateways	—
RQA_4 / RQB_4	S3.M2 FireControl	S3.M5 DamageCalc
RQC_4	S3.M2 FireControl	S1.M1 Input
RQD_4	S3.M2 FireControl	—
RQA_5	S3.M1 AimReticle	—
RQB_5	S3.M1 AimReticle	S1.M1 Input
RQC_5	S3.M1 AimReticle	S3.M2 FireControl
RQA_6 / RQB_6	S3.M3 AmmoManager	S3.M2 FireControl
RQC_6 / RQD_6	S3.M4 Reloading	S1.M1 Input; S3.M3 AmmoManager
RQE_6	S3.M3 AmmoManager	HUD
RQA_7(Salud)/RQB_7(Salud)	S3.M5 DamageCalc	HUD
RQC_7(Salud)	Pantalla Derrota	S2.M3 SceneState
RQA_8(Inv)/RQB_8(Inv)	S5.M1 Inventory	S5.M2 LootPickup
RQC_8(Inv)	S5.M1 Inventory	—
RQD_8(Inv)	S5.M1 Inventory	—
RQA_9(Recursos)	S5.M3 LootSpawn	S2.M1 MapLoader
RQB_9(Recursos)	S5.M2 LootPickup	S5.M1 Inventory
RQC_9(Recursos)	S5.M3 LootSpawn	—
RQA_10 / RQB_10	S3.M3 AmmoManager	S3.M2 FireControl
RQA_7(Enemigos)	S4.M1 Spawner	S2.M1 MapLoader

RQB_7(Enemigos)	S4.M1 Spawner	—
RQC_7(Enemigos)	S4.M1 Spawner	—
RQA_8(Enemigos)	S4.M2 EnemyAI	—
RQB_8(Enemigos)	S4.M3 EnemyAttack	S3.M5 DamageCalc
RQC_8(Enemigos)	S4.M3 EnemyAttack	—
RQA_9(UiMenu)	Pantalla Menú	—
RQA_10(HUD)	HUD	S3.M3 AmmoManager; S5.M1 Inventory; S3.M5 DamageCalc
RQA_11	Pantalla Derrota	S2.M3 SceneState
RQA_12	Pantalla Victoria	S2.M3 SceneState
RQA_13 / RQB_14	S6.M1 AchievementsBST	S6.M2 AchievementsView
RQA_15	Sistema de Diálogos	—
RQA_16	Animaciones	—
RQA_17	Manual	—