

TwinSight X500: prototipo web 3D para visualización técnica de hardware complejo

Estudiante

Alexander Woodcock Salomón

Asesor

Gustavo Enrique Vejarano Matiz

Universidad Nacional Abierta y a Distancia - UNAD

Escuela de Ciencias Básicas, Tecnología e Ingeniería

Ingeniería Multimedia

Pasto, Colombia

2026

Resumen

La documentación de hardware técnico complejo, como drones y sistemas electromecánicos multicomponente, presenta limitaciones de comunicación espacial cuando depende de imágenes 2D, manuales PDF y fuentes técnicas fragmentadas. Este informe desarrolla TwinSight X500, un prototipo web 3D interactivo orientado a la visualización técnica, inspección y análisis visual del dron Holybro X500 V2. El trabajo se enmarca en Design Science Research, prototipado iterativo, Teoría de la Carga Cognitiva, interacción 3D y renderizado físicamente basado. La build integra selección jerárquica de piezas, ficha técnica inferior, modos *Inspect*, *Analyze* y *Studio*, corte transversal, vista explosionada, filtros y visualización térmica heurística. El pipeline redujo el modelo desde rutas CAD superiores a 6,5 millones de triángulos hasta 95 617 triángulos optimizados. La validación combinó profiler WebGL, desempeño en tareas cronometradas, SUS, NASA-TLX Raw y *Think-Aloud* con 12 participantes: SUS promedio de 91,88, NASA-TLX Raw de 8,69 para el visor 3D frente a 19,89 para el soporte 2D, menor tiempo medio en las tres tareas cronometradas y 96 registros tarea-condición completados. El prototipo alcanzó rendimiento estable en escritorio, comportamiento funcional en móviles reales y alcance defendible como *visual product twin*, no como gemelo digital operacional.

Palabras clave: WebGL, Unity, WebAssembly, visualización técnica 3D, hardware complejo, visual product twin, carga de trabajo percibida.

Abstract

Complex hardware documentation, including drones and multicomponent electromechanical systems, presents spatial communication limits when it depends on 2D images, PDF manuals, and fragmented technical sources. This report develops TwinSight X500, an interactive 3D web prototype for technical visualization, inspection, and visual analysis of the Holybro X500 V2. The work is framed through Design Science Research, iterative prototyping, cognitive load theory, 3D interaction, and physically based rendering. The build includes hierarchical part selection, a bottom technical sheet, *Inspect*, *Analyze*, and *Studio* modes, cross-section, exploded view, category filters, and component-based heuristic thermal visualization. The pipeline reduced the model from CAD routes above 6.5 million triangles to 95,617 optimized triangles. Validation combined WebGL profiling, timed task performance, SUS, NASA-TLX Raw, and Think-Aloud with 12 participants: mean SUS was 91.88, NASA-TLX Raw was 8.69 for the 3D viewer versus 19.89 for the 2D reference, lower mean time in the three timed tasks for the 3D condition, and 96 completed task-condition records. The prototype achieved stable desktop performance, functional behavior on real mobile devices, and a defensible scope as a visual product twin, not as an operational digital twin.

Keywords: WebGL, Unity, WebAssembly, technical 3D visualization, complex hardware, visual product twin, perceived workload.

Tabla de Contenido

Resumen	2
Abstract	3
Información General del Trabajo de Grado	18
Integrantes del Proyecto	18
Datos Específicos del Proyecto	18
Introducción	19
Planteamiento del problema	21
Justificación	23
Trade-off explícito entre tooling y huella inicial	25
Ejecución de lógica compilada en WebAssembly	25
<i>Pipeline</i> integrado de importación y renderizado	25
Capacidad de extensión para visualización técnica	25
Compatibilidad esperada de la plataforma web	26
Aporte a la Ingeniería Multimedia	26
Objetivos	27
Objetivo general	27
Objetivos específicos	27
Alcance y limitaciones	28
Marco de Referencia	31
Estado del Arte	31
Digital twins, digital shadows e interoperabilidad industrial	31
Benchmarking de soluciones Web 3D	35

Three.js	35
Babylon.js	35
Unity Web	35
Unreal Engine Pixel Streaming	36
Spline	36
Marmoset Viewer	36
Sketchfab	36
PlayCanvas	37
Brecha identificada y selección de Unity Web	37
Marco Teórico	38
Madurez de integración digital en hardware complejo	39
Teoría de la Carga Cognitiva (Sweller, 1988; Sweller et al., 2019)	39
Carga intrínseca	41
Carga extrínseca	41
Carga germana	41
Navegación espacial e interacción 3D	41
Cognición distribuida (Hutchins, 1995)	43
Heurísticas de usabilidad (Nielsen, 1994)	43
Rotación mental y visualización espacial	44
Percepción visual y organización de la información	44
Fundamentos matemáticos de optimización gráfica	44
Presupuesto poligonal	46
Densidad de texel	46

Costo de dibujo y agrupación de mallas	46
<i>Frame time</i> y objetivo de FPS.	47
Modelo matemático de renderizado físicamente basado (PBR)	48
BRDF especular de Cook-Torrance	49
Aproximación de Fresnel de Schlick	49
Distribución GGX	50
Conservación de energía	50
Relación con la app real	50
Iluminación basada en imágenes (IBL)	51
Matemática aplicada a shaders de inspección y análisis	51
Plano de corte global	51
Fresnel de borde para modos analíticos	54
Mapeo visual del modo térmico	54
Marco Conceptual	56
Marco Metodológico	61
Tipo de investigación y enfoque	61
Marco de Design Science Research	62
Delimitación metodológica como visual product twin	63
Fases de trabajo	65
Prueba breve formativa previa a la evaluación final	65
Muestra y perfil de participantes	67
Variables de investigación	69
Variable independiente	69

Variables dependientes	69
Variables de control y condiciones registradas	72
Diseño comparativo y condiciones de prueba	73
Definición operativa de la condición 2D	73
Instrumentos	75
Protocolo de tareas	77
Procedimiento de análisis	77
Análisis cuantitativo	77
Análisis cualitativo	79
Triangulación de hallazgos	79
Consideraciones éticas	80
Desarrollo e Implementación	82
Estado del desarrollo	82
Normalización del sistema	83
Evolución del proyecto y consolidación del caso de estudio	86
Lectura del prototipo como visual product twin	87
Pipeline 3D, importación CAD y saneamiento geométrico	89
Métodos de importación CAD y criterio de uso	89
MAD-T como marco técnico adaptado para activos hard-surface WebGL	89
Automatización propia en Blender: masters, instancias y proxies	96
Sistema modular de tornillería y ahorro geométrico	97
Software especializado y costo del pipeline	103
Automatización y addons creados para el proyecto	105

Evolución de la estrategia de LOD y migración técnica evaluada	110
UX/UI, diseño visual y enfoque <i>mobile-first</i>	110
Ficha inferior, panel de información y jerarquía de selección	114
Sistema de iconos procedurales y microinteracciones	118
Onboarding procedural y ayuda contextual	119
Arquitectura operativa	120
Bootstrap y saneamiento del import	128
Garantía de managers	128
Reparación del modelo importado	128
Flujo visible de la build final	129
Hero	129
Onboarding y primer uso	129
Explore y selección	130
Inspect	130
Analyze	130
Studio	130
Sistema de visualización, shaders y entornos	139
Modo Blueprint y lectura analítica	141
Entornos, presets y color base	142
Modelo matemático operativo del subsistema térmico	142
Matriz de estado funcional	149
Expuesto en UI final	149
Implementado pero oculto	151

Código legado o no integrado	151
Tooling de editor, automatización y verificación	152
Limitaciones y frentes de cierre	155
Resultados y Análisis	158
Estado de cierre funcional	158
Aporte frente a necesidades de la industria de hardware complejo	158
Resultados formativos de rendimiento	162
Ajustes formativos previos a la evaluación con usuarios	168
Resultados de usabilidad	171
Resultados de carga de trabajo percibida	177
Resultados cualitativos	179
Resultados del pipeline de activos	182
Nota de cierre	186
Discusión	186
Conclusiones	189
Respecto al objetivo general	189
Respecto a los objetivos específicos	190
OE1: pipeline de optimización y preparación de activos	190
OE2: materiales, modos de visualización y rendimiento WebGL	190
OE3: implementación del prototipo interactivo	190
OE4: evaluación formativa 3D vs. 2D, usabilidad y carga de trabajo	191
Limitaciones	191
Trabajo futuro	193

Recomendaciones	194
Referencias	196
Apéndices	204
Apéndice A: Repositorio de código fuente	204
Apéndice B: Manual de usuario	204
Apéndice C: Manual técnico	204
Apéndice D: Documentación técnica activa	205
Apéndice E: Matriz de desconexiones	205
Apéndice F: Catálogo canónico del Holybro X500 V2	205
Apéndice G: Validación funcional de la escena final	206
Apéndice H: Instrumentos y registros de evaluación	206
Apéndice I: Estado del pipeline de activos	210
Apéndice J: Demo público	211
Apéndice K: Audio y alcance no integrado	211
Apéndice L: Auditoría matemática y trazabilidad con Wolfram	212
Apéndice M: Correlación entre la propuesta y el informe final	212
Apéndice N: Resumen de artefactos activos del cierre	214

Lista de Tablas

<i>Tabla 1</i>	<i>Comparativa cualitativa de soluciones Web 3D</i>	<i>37</i>
<i>Tabla 2</i>	<i>Perfil académico o profesional agregado de la muestra</i>	<i>69</i>
<i>Tabla 3</i>	<i>Reconciliación entre conteos históricos y convenciones finales del proyecto</i>	<i>85</i>
<i>Tabla 4</i>	<i>Hitos de evolución técnica del desarrollo del proyecto</i>	<i>88</i>
<i>Tabla 5</i>	<i>Métodos de importación CAD considerados en el proyecto</i>	<i>90</i>
<i>Tabla 6</i>	<i>Trazabilidad CAD–runtime</i>	<i>95</i>
<i>Tabla 7</i>	<i>Criterios para comparar el saneamiento geométrico</i>	<i>98</i>
<i>Tabla 8</i>	<i>Lectura técnica del sistema modular de fasteners</i>	<i>100</i>
<i>Tabla 9</i>	<i>Software y addons del pipeline 3D</i>	<i>105</i>
<i>Tabla 10</i>	<i>Automatizaciones y addons asociados al pipeline</i>	<i>108</i>
<i>Tabla 11</i>	<i>Criterios de diseño aplicados a la UI del prototipo</i>	<i>114</i>
<i>Tabla 12</i>	<i>Modos visuales, publicación en UI y lectura técnica</i>	<i>143</i>
<i>Tabla 13</i>	<i>Inventario sintético de módulos y estado de integración</i>	<i>159</i>
<i>Tabla 14</i>	<i>Estado de cierre funcional por frente de validación</i>	<i>159</i>
<i>Tabla 15</i>	<i>Relación entre necesidades industriales y respuesta del prototipo</i>	<i>160</i>
<i>Tabla 16</i>	<i>Resumen de rendimiento WebGL por dispositivo y escenario operativo . . .</i>	<i>163</i>
<i>Tabla 17</i>	<i>KPIs complementarios de build, memoria y escena runtime</i>	<i>164</i>
<i>Tabla 18</i>	<i>Detalle de sesiones de rendimiento WebGL por dispositivo y escenario . .</i>	<i>164</i>
<i>Tabla 19</i>	<i>Compatibilidad por factor de forma, navegador y estado de validación . .</i>	<i>167</i>

<i>Tabla 20</i>	<i>Hallazgos formativos y correcciones</i>	<i>170</i>
<i>Tabla 21</i>	<i>Desempeño observado por tarea y condición</i>	<i>172</i>
<i>Tabla 22</i>	<i>Tiempos registrados por participante en tareas cronometradas</i>	<i>173</i>
<i>Tabla 23</i>	<i>Resumen descriptivo de tiempos en tareas cronometradas</i>	<i>173</i>
<i>Tabla 24</i>	<i>Resultados SUS y NASA-TLX Raw por participante</i>	<i>175</i>
<i>Tabla 25</i>	<i>Resumen descriptivo de usabilidad y carga de trabajo percibida</i>	<i>176</i>
<i>Tabla 26</i>	<i>Promedios NASA-TLX Raw por dimensión y condición</i>	<i>178</i>
<i>Tabla 27</i>	<i>Síntesis de hallazgos Think-Aloud</i>	<i>181</i>
<i>Tabla 28</i>	<i>Conteo total de triángulos por método de importación y optimización . . .</i>	<i>183</i>
<i>Tabla 29</i>	<i>Matriz de equivalencia informativa entre visor 3D y soporte 2D</i>	<i>207</i>
<i>Tabla 30</i>	<i>Estado del pipeline 3D al cierre documental</i>	<i>211</i>
<i>Tabla 31</i>	<i>Correlación entre tablas de la propuesta y el informe final</i>	<i>213</i>
<i>Tabla 32</i>	<i>Productos entregados vs. esperados al cierre documental</i>	<i>213</i>
<i>Tabla 33</i>	<i>Resumen de artefactos activos del cierre documental</i>	<i>214</i>

Lista de Figuras

<i>Figura 1</i>	<i>Fragmentación de información en hardware complejo.</i>	<i>20</i>
<i>Figura 2</i>	<i>Dimensiones del núcleo problemático del proyecto.</i>	<i>22</i>
<i>Figura 3</i>	<i>Alcance actual del proyecto frente a un gemelo digital operacional.</i>	<i>24</i>
<i>Figura 4</i>	<i>Escalera de madurez desde digital model hasta digital twin operacional.</i>	<i>33</i>
<i>Figura 5</i>	<i>Interoperabilidad futura mediante twin manifest por componente.</i>	<i>34</i>
<i>Figura 6</i>	<i>Capas del prototipo como modelo visual-semántico de producto.</i>	<i>40</i>
<i>Figura 7</i>	<i>Relación entre documentación 2D, carga cognitiva y apoyo visual 3D.</i>	<i>42</i>
<i>Figura 8</i>	<i>Balance entre fidelidad visual, legibilidad técnica y rendimiento WebGL.</i>	<i>45</i>
<i>Figura 9</i>	<i>Modelo operativo de rendimiento WebGL.</i>	<i>48</i>
<i>Figura 10</i>	<i>Lectura visual del flujo PBR usado por URP en la app.</i>	<i>51</i>
<i>Figura 11</i>	<i>Proceso de corte transversal aplicado por shader.</i>	<i>53</i>
<i>Figura 12</i>	<i>Intensidad Fresnel y contornos técnicos en sombreado de borde.</i>	<i>54</i>
<i>Figura 13</i>	<i>Jerarquía térmica heurística por componentes.</i>	<i>55</i>
<i>Figura 14</i>	<i>Aplicación del ciclo Design Science Research al proyecto WebGL.</i>	<i>63</i>
<i>Figura 15</i>	<i>Frontera metodológica entre lo evaluado y lo proyectado.</i>	<i>64</i>
<i>Figura 16</i>	<i>Fases de trabajo del proyecto.</i>	<i>66</i>
<i>Figura 17</i>	<i>Ciclo de QA formativo y optimización de interacción.</i>	<i>68</i>
<i>Figura 18</i>	<i>Diseño comparativo 3D frente a 2D</i>	<i>75</i>
<i>Figura 19</i>	<i>Instrumentos de evaluación del prototipo.</i>	<i>76</i>

<i>Figura 20</i>	<i>Triangulación de evidencia para interpretar el prototipo.</i>	<i>80</i>
<i>Figura 21</i>	<i>Taxonomía 28/30/257 del sistema</i>	<i>86</i>
<i>Figura 22</i>	<i>Flujo de decisión para importación y optimización CAD a WebGL.</i>	<i>90</i>
<i>Figura 23</i>	<i>Importación directa STEP a Blender con vista sombreada y wireframe.</i>	<i>91</i>
<i>Figura 24</i>	<i>Importación desde MoI3D sin triangulación final con vista sombreada y wireframe.</i>	<i>91</i>
<i>Figura 25</i>	<i>Adaptación del flujo MAD-T al pipeline CAD–WebGL del proyecto.</i>	<i>94</i>
<i>Figura 26</i>	<i>Tooling propio en Blender para masters, instancias, proxies y manifest.</i>	<i>97</i>
<i>Figura 27</i>	<i>Saneamiento de geometría CAD para runtime WebGL.</i>	<i>98</i>
<i>Figura 28</i>	<i>Piezas base del sistema modular de tornillos.</i>	<i>101</i>
<i>Figura 29</i>	<i>Ejemplo de tornillo generado mediante arreglo modular.</i>	<i>102</i>
<i>Figura 30</i>	<i>Sistema modular delimitado a tornillos y reconciliación de fasteners.</i>	<i>102</i>
<i>Figura 31</i>	<i>Fasteners originales y proxies ligeros usados en el pipeline.</i>	<i>103</i>
<i>Figura 32</i>	<i>Software auxiliar para baking, lookdev y despliegue UV.</i>	<i>106</i>
<i>Figura 33</i>	<i>Tooling propio para automatizar el pipeline CAD.</i>	<i>109</i>
<i>Figura 34</i>	<i>Modelo de alta resolución usado como fuente de bake.</i>	<i>111</i>
<i>Figura 35</i>	<i>Malla optimizada del dron después del cierre geométrico.</i>	<i>112</i>
<i>Figura 36</i>	<i>Evolución de la estrategia de LOD</i>	<i>113</i>
<i>Figura 37</i>	<i>Evolución de la interfaz hacia el enfoque mobile-first.</i>	<i>115</i>
<i>Figura 38</i>	<i>Hero móvil y adaptación funcional a escritorio.</i>	<i>115</i>

<i>Figura 39</i>	<i>Explore móvil y adaptación funcional a escritorio.</i>	116
<i>Figura 40</i>	<i>Acceso a información de pieza y panel contextual.</i>	117
<i>Figura 41</i>	<i>Iconos procedurales y microinteracciones</i>	119
<i>Figura 42</i>	<i>Onboarding procedural de primer uso integrado al runtime.</i>	121
<i>Figura 43</i>	<i>Tarjetas del onboarding procedural: navegación, selección e información.</i>	122
<i>Figura 44</i>	<i>Tarjetas del onboarding procedural: inspección, hotspots y aislamiento.</i>	122
<i>Figura 45</i>	<i>Tarjetas del onboarding procedural: energía, análisis y corte.</i>	123
<i>Figura 46</i>	<i>Tarjetas del onboarding procedural: explosión, filtros y Studio.</i>	123
<i>Figura 47</i>	<i>Tarjetas del onboarding procedural: modo de render, ambiente e iluminación.</i>	124
<i>Figura 48</i>	<i>Arquitectura por capas del sistema.</i>	125
<i>Figura 49</i>	<i>Flujo de comunicación EventBus bajo el patrón publish/subscribe.</i>	126
<i>Figura 50</i>	<i>Máquina de estados de la aplicación (AppStateMachine).</i>	127
<i>Figura 51</i>	<i>Flujo de datos DronePartData</i>	129
<i>Figura 52</i>	<i>Menús principales de la barra inferior de la app.</i>	131
<i>Figura 53</i>	<i>Flujo de selección de piezas desde input y raycast hasta la interfaz.</i>	132
<i>Figura 54</i>	<i>Hero móvil del visor en inglés y español.</i>	133
<i>Figura 55</i>	<i>Pantallas auxiliares del Hero.</i>	134
<i>Figura 56</i>	<i>Controles de energía y corte transversal.</i>	134
<i>Figura 57</i>	<i>Controles de vista explosionada y filtros.</i>	135

<i>Figura 58</i>	<i>Panel de control del menú Studio.</i>	135
<i>Figura 59</i>	<i>Hotspot y secuencia de selección en Explore.</i>	136
<i>Figura 60</i>	<i>Secuencia de aislamiento por niveles en Inspect.</i>	137
<i>Figura 61</i>	<i>Efectos visibles de Explode, Cut y filtros sobre el modelo.</i>	138
<i>Figura 62</i>	<i>Mapa de modos visuales por propósito y lugar en la UI.</i>	140
<i>Figura 63</i>	<i>Pipeline de shaders por modo de visualización</i>	141
<i>Figura 64</i>	<i>Modos directos de render para inspección visual del ensamblaje.</i>	144
<i>Figura 65</i>	<i>Presets de Studio y lectura Blueprint.</i>	145
<i>Figura 66</i>	<i>Variación de entorno y color base desde Studio.</i>	146
<i>Figura 67</i>	<i>Arquitectura del subsistema térmico híbrido.</i>	149
<i>Figura 68</i>	<i>Lectura visual del modo Thermal en la build final.</i>	150
<i>Figura 69</i>	<i>Coherencia documental entre app, manuales e informe.</i>	152
<i>Figura 70</i>	<i>Restricciones de WebGL y mitigaciones técnicas aplicadas al proyecto.</i>	154
<i>Figura 71</i>	<i>Flujo de despliegue y artefactos desde la build hasta el hosting y QA.</i>	155
<i>Figura 72</i>	<i>Estado de herramientas de ensamblaje</i>	157
<i>Figura 73</i>	<i>Necesidades industriales, respuesta actual y evolución futura defendible.</i>	161
<i>Figura 74</i>	<i>Matriz visual de compatibilidad por factor de forma y dispositivo.</i>	168
<i>Figura 75</i>	<i>Mapa de gestos espontáneos y correcciones P1/P2.</i>	171
<i>Figura 76</i>	<i>Distribución del puntaje SUS en la muestra evaluada.</i>	177
<i>Figura 77</i>	<i>Comparativa NASA-TLX Raw por dimensiones.</i>	179

<i>Figura 78</i>	<i>Mapa de codificación cualitativa del protocolo Think-Aloud.</i>	<i>182</i>
<i>Figura 79</i>	<i>Malla optimizada final y lectura de wireframe.</i>	<i>184</i>
<i>Figura 80</i>	<i>Renders del dron optimizado con mapas de normales y oclusión ambiental.</i>	<i>185</i>
<i>Figura 81</i>	<i>Roadmap visual de evolución desde visual product twin hasta digital twin operacional.</i>	<i>194</i>
<i>Figura 82</i>	<i>Muestra de una captura del profiler interno.</i>	<i>209</i>

Información General del Trabajo de Grado

Integrantes del Proyecto

Nombre del estudiante: Alexander Woodcock Salomón

Programa Académico: Ingeniería Multimedia

% de créditos aprobados: 100

Municipio / Departamento: Pasto / Nariño

Datos de contacto e identificación: omitidos en esta copia pública por privacidad.

Datos Específicos del Proyecto

Duración del proyecto: Un (1) año

Línea de Investigación: Ingeniería Multimedia, Visualización Técnica 3D Web

Escuela: Escuela de Ciencias Básicas, Tecnología e Ingeniería (ECBTI)

Descriptor palabras clave: WebGL, Unity, WebAssembly, visualización técnica 3D, optimización, carga de trabajo percibida, HCI, Technical Art, hardware complejo, visual product twin, digital shadow.

Introducción

La documentación de hardware técnico complejo —como drones y otros sistemas electromecánicos multicomponente— enfrenta un desafío de comunicación espacial cuando depende de imágenes 2D, planos convencionales y documentos PDF estáticos. A lo largo de este trabajo, el término *hardware complejo* designa sistemas con integración estructural, energética, electrónica y de control, alta densidad de relaciones espaciales entre piezas y necesidad de inspección visual detallada del ensamblaje. Bajo estas condiciones, los medios estáticos obligan a ingenieros y técnicos a reconstruir mentalmente relaciones tridimensionales que no se presentan de forma explícita, lo que incrementa la demanda cognitiva asociada a la tarea y dificulta la lectura técnica del sistema (Sweller, 1988; Hegarty & Waller, 2004).

Esta dificultad no es exclusiva del sector de drones. En industrias que diseñan, ensamblan, operan o mantienen hardware complejo, la geometría CAD, las listas de materiales, los manuales, la telemetría, los historiales de mantenimiento y el conocimiento experto suelen encontrarse distribuidos en sistemas separados. La literatura sobre *digital twins* identifica esa fragmentación y la falta de interoperabilidad como barreras recurrentes para convertir representaciones digitales en herramientas útiles durante el ciclo de vida del producto (Jones et al., 2020; Lin et al., 2023). Por ello, antes de aspirar a un gemelo digital operacional, muchas organizaciones requieren una capa previa: un modelo visual-semántico confiable que permita comprender, inspeccionar y preparar el producto para futuras integraciones de datos (véase la Figura 1).

La convergencia entre WebAssembly, WebGL y *pipelines* contemporáneos de renderizado físicamente basado abrió una oportunidad concreta para trasladar parte de esa interpretación espacial al propio sistema interactivo (Yu et al., 2023). Sobre esa base, el proyecto se orientó al

Figura 1

Fragmentación de información en hardware complejo.



Nota. El diagrama sintetiza la separación habitual entre CAD, BOM, manuales, sensores, mantenimiento y conocimiento experto como punto de partida para una capa visual-semántica. Elaboración propia.

diseño, desarrollo y documentación de un prototipo web 3D interactivo para la visualización técnica, inspección y análisis visual del ensamblaje del dron Holybro X500 V2, buscando un equilibrio entre claridad técnica, rendimiento web y coherencia de experiencia de usuario.

Planteamiento del problema

La migración de documentación técnica a entornos 3D interactivos no depende solo de una mejor presentación visual. También exige resolver restricciones reales de despliegue web. Los modelos CAD de ingeniería, basados en NURBS, ensamblajes extensos y activos de alta densidad geométrica, no son directamente compatibles con las condiciones de ejecución de una aplicación web. Si bien WebAssembly permite el crecimiento dinámico de la memoria lineal, la viabilidad práctica del producto depende de dos factores: controlar la huella inicial de los activos y configurar un *heap* contiguo lo suficientemente grande para sostener datos, geometría y ejecución estable en el navegador (Unity Technologies, 2024a; WebAssembly Community Group, s. f.).

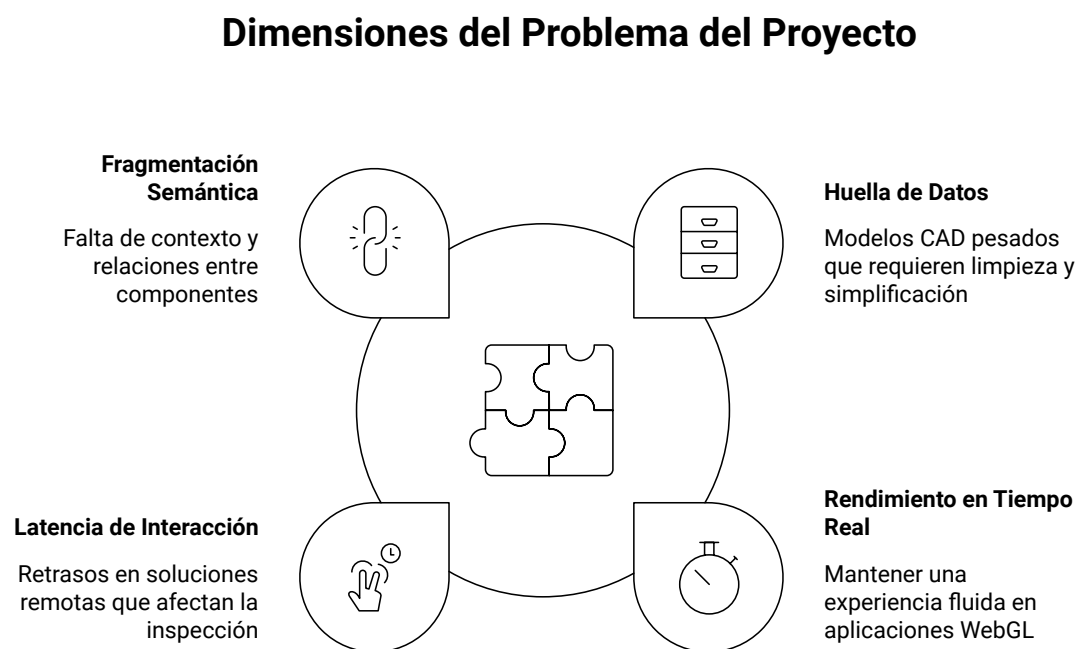
En este proyecto, el problema técnico se articuló en cuatro dimensiones complementarias (véase la Figura 2):

- **Huella inicial de datos.** Los modelos CAD originales pueden alcanzar decenas o cientos de megabytes, e incluso más en ensamblajes extensos, por lo que requieren limpieza, simplificación geométrica, *baking* y empaquetado selectivo para ser viables en web.
- **Rendimiento en tiempo real.** Mantener 30 FPS o más, equivalente a un *frame time* de aproximadamente 33,33 ms, exige controlar presupuesto geométrico, materiales, *draw calls* y estados de render.

- **Latencia de interacción.** Alternativas como *Pixel Streaming* pueden ofrecer render remoto de alta calidad visual, pero introducen dependencia de infraestructura, red y latencias poco adecuadas para inspección técnica precisa (Epic Games, s. f.-a, s. f.-b).
- **Fragmentación de datos técnicos.** La geometría del producto no basta para explicar su función, mantenimiento o evolución. Para que una representación digital avance hacia *digital shadow* o *digital twin*, debe existir una relación trazable entre piezas, metadatos, procedimientos, sensores, eventos y decisiones de soporte (Kritzinger et al., 2018; Digital Twin Consortium, 2020).

Figura 2

Dimensiones del núcleo problemático del proyecto.



Nota. La figura resume los cuatro frentes que condicionan la viabilidad del prototipo: huella inicial de datos, rendimiento en tiempo real, latencia de interacción y fragmentación semántica. Elaboración propia.

Se identificó, por tanto, la necesidad de documentar un *pipeline* de *Technical Art* que permitiera desplegar hardware complejo en un visor 3D web con orientación espacial suficiente para tareas de inspección, costo computacional controlado y capacidad de análisis visual del ensamblaje para propósitos académicos e ingenieriles.

El proyecto se orientó por dos preguntas de investigación complementarias:

1. **¿Qué diferencias descriptivas se observan en el desempeño de tareas y la carga de trabajo percibida cuando usuarios con perfil afín realizan actividades de exploración e identificación del dron Holybro X500 V2 mediante un prototipo web 3D interactivo, en comparación con un soporte 2D de referencia? Adicionalmente, ¿bajo qué condiciones de rendimiento y memoria resulta técnicamente viable dicho prototipo en el navegador?**
2. **¿Cuál es el nivel de usabilidad percibida del prototipo 3D medido con SUS tras su uso?**

Justificación

La elección de Unity Web frente a bibliotecas nativas de JavaScript o soluciones de *streaming* remoto no obedeció a un menor tamaño de *build*. La decisión se sustentó en la integración del *pipeline*, la capacidad de extensión técnica, el soporte para materiales y shaders, y la posibilidad de mantener un flujo coherente entre programación, arte técnico y evaluación del prototipo.

Desde la perspectiva industrial, el valor del proyecto radica en cerrar una primera brecha de madurez digital: transformar un producto electromecánico multicomponente en una experiencia web navegable, seleccionable, explicable y medible. En este sentido, el prototipo funciona como un *visual product twin* o *technical product twin*: una capa visual-semántica que todavía no

sincroniza telemetría real, pero sí prepara identificadores, jerarquías, categorías, fichas y herramientas de inspección que podrían alimentar fases posteriores de *digital shadow* y *digital twin* operacional (Kritzinger et al., 2018; Digital Twin Consortium, 2020; Lin et al., 2023; véase la Figura 3).

Figura 3

Alcance actual del proyecto frente a un gemelo digital operacional.

¿Cómo se debe clasificar la versión actual del proyecto?

Gemelo de Producto Visual			Gemelo Digital Operacional	
Permite la visualización técnica y la preparación semántica de hardware complejo. Se centra en la inspección, selección y medición.		Funcionalidad Principal	 Requiere datos en tiempo real, validación e integración con sistemas externos. Se centra en la sincronización operativa y la telemetría en vivo.	
Representa el estado implementado del proyecto. Validado como el posicionamiento académico correcto para la versión actual.		Estado Actual del Desarrollo	 Representa la hoja de ruta futura del proyecto. Considerado un término no defendible para la versión actual.	
Utiliza datos semánticos estáticos para la identificación de piezas. Se basa en taxonomías preparadas para el aislamiento de componentes.		Requisitos de Datos	 Exige flujos de datos en vivo y modelos calibrados. Depende de bucles de retroalimentación continuos e integración de sistemas.	

Nota. La versión final se posiciona como *visual product twin*: integra inspección, selección y preparación semántica, pero no telemetría real, sincronización operacional ni modelos calibrados. Elaboración propia.

Trade-off explícito entre tooling y huella inicial

Unity no fue seleccionado por ofrecer la descarga inicial más ligera. Por el contrario, el proyecto aceptó una huella inicial mayor a cambio de un ecosistema de autoría más consistente: editor integrado, *profiling*, flujo de materiales, UI Toolkit, sistema de escenas y facilidad para extender el comportamiento mediante C# y shaders personalizados (Unity Technologies, 2024b, 2024d).

Ejecución de lógica compilada en WebAssembly

En el target web, Unity convierte código intermedio de .NET a C++ mediante IL2CPP y posteriormente compila ese resultado a WebAssembly usando el *toolchain* del *build*. Esto permitió sostener una base tipada para la lógica interactiva del prototipo y mantener el proyecto dentro de un entorno técnico unificado, sin atribuir a IL2CPP, por sí solo, todo el salto a Wasm (Unity Technologies, 2024c; Unity Technologies, s. f.).

Pipeline integrado de importación y renderizado

El proyecto exigía coordinar limpieza de activos CAD, materiales PBR, UI responsiva, modos analíticos y validación funcional sin fragmentar el flujo entre demasiadas herramientas. Unity aportó un entorno integrado para importar geometría, gestionar materiales, perfilar rendimiento y desplegar un *build* web desde una misma base de trabajo.

Capacidad de extensión para visualización técnica

La aplicación final no se concibió como un visor pasivo de un modelo tridimensional, sino como una experiencia de inspección técnica. El usuario puede recorrer el ensamblaje, seleccionar componentes, aislar zonas de interés, consultar información contextual, separar visualmente el conjunto para comprender relaciones de montaje, observar cortes del interior y alternar lec-

turas visuales que resaltan materialidad, estructura o comportamiento térmico aproximado. Estas capacidades exigían una plataforma capaz de articular renderizado realista, interacción con piezas, organización semántica del hardware y herramientas de análisis visual dentro de un mismo producto.

Compatibilidad esperada de la plataforma web

La plataforma web de Unity contempla navegadores compatibles de escritorio y algunos navegadores móviles, aunque el comportamiento efectivo depende del navegador, su versión, la memoria disponible y el nivel de optimización del *build* (Unity Technologies, s. f.). Por ello, en este trabajo la compatibilidad móvil se documenta como *compatibilidad esperada*, sin pretender constituir una garantía universal ni una validación exhaustiva sobre una matriz cerrada de dispositivos.

Aporte a la Ingeniería Multimedia

El proyecto aporta al programa por al menos cuatro vías:

- documenta un *pipeline* técnico para llevar activos CAD a un visor web interactivo;
- articula computación gráfica, experiencia de usuario, documentación técnica y arte técnico en un mismo caso aplicado;
- convierte restricciones de rendimiento web en criterios medibles de diseño y optimización;
- consolida un caso de estudio pertinente para la Ingeniería Multimedia, donde la claridad técnica del ensamblaje es tan importante como la calidad visual del resultado.

Objetivos

Objetivo general

Desarrollar un prototipo de visualización web 3D interactiva basado en Unity Web, enfocado en la exploración técnica, la inspección y el análisis visual del ensamblaje de hardware complejo, optimizado mediante criterios de *Technical Art* para su despliegue eficiente en navegadores web compatibles.

Objetivos específicos

1. Diseñar un *pipeline* de optimización de activos 3D que reduzca la carga poligonal de modelos CAD originales hasta un presupuesto geométrico compatible con WebGL, preservando la legibilidad formal del ensamblaje, la identificación correcta de piezas y la lectura espacial suficiente para inspección técnica mediante retopología, limpieza de mallas y *baking* de texturas.
2. Integrar en Unity URP un sistema de materiales y modos de visualización que combine renderizado realista con herramientas de inspección técnica, procurando un *frame time* igual o inferior a 33,33 ms (30 FPS o más) bajo el entorno de prueba controlado definido para la validación del *build* web.
3. Implementar el prototipo interactivo en la plataforma web de Unity con lógica desarrollada en C# y desplegada mediante la cadena de compilación del motor — conversión de ensamblados administrados, generación intermedia de C++ y ejecución en WebAssembly—, de modo que habilite navegación orbital, selección de piezas, consulta contextual de información y herramientas como vista explosionada, corte transversal y modos analíticos.

4. Evaluar de manera formativa el desempeño en tareas y la carga de trabajo percibida por condición (3D vs. 2D) mediante pruebas con usuarios, NASA-TLX Raw y protocolo *Think-Aloud*, y estimar adicionalmente la usabilidad percibida del prototipo 3D mediante SUS tras su uso; todo ello contrastado de forma descriptiva y frente a las metas operativas definidas para el proyecto.

Alcance y limitaciones

El alcance del presente trabajo se circunscribió al desarrollo de un **Producto Mínimo Viable** (MVP) técnico y académico orientado a documentar la viabilidad de un visor web 3D interactivo para hardware complejo. El caso de estudio seleccionado fue el dron Holybro X500 V2 porque cuenta con recursos técnicos de libre acceso, archivos CAD/STEP consultables, documentación pública y abundantes referencias abiertas en internet. Esta disponibilidad permitió ejecutar el proyecto sin depender de material propietario restringido, sin comprometer derechos de autor y con suficiente trazabilidad para justificar decisiones de modelado, documentación e interpretación técnica.

Las principales delimitaciones fueron las siguientes:

- **Modelo único.** El prototipo se desarrolló sobre un único modelo de referencia, aunque la arquitectura quedó preparada para generalizar catálogos, datos y saneamiento de runtime.
- **No constituye un digital twin operacional.** La versión final no recibe telemetría real, no sincroniza estado con un activo físico, no ejecuta diagnóstico de mantenimiento ni se integra con sistemas PLM, ERP, CMMS, SCADA o IoT. Su alcance correcto es el de una plataforma visual-semántica preparada para evolucionar hacia esas capas.

- **Plataforma web.** El producto se concentró en la plataforma web compilada con WebAssembly. No se desarrollaron *builds* nativas independientes de escritorio o móvil.
- **Compatibilidad móvil acotada.** El producto se concibió para navegadores compatibles de escritorio y móviles. En el cierre de desarrollo se verificó su funcionamiento en escritorio y en dispositivos móviles reales, incluyendo teléfonos de gama media-baja como Redmi Note 10S. La compatibilidad se interpreta dentro de las condiciones de prueba de cada dispositivo, navegador, resolución y escenario, sin asumir cobertura universal sobre todo el ecosistema móvil.
- **Muestra de validación.** Se estableció como meta deseable un grupo de 30 participantes y como escenario mínimo operativo una muestra de entre 8 y 12 para evaluación formativa. El informe integra datos de 12 participantes anonimizados, con SUS, NASA-TLX Raw, tareas completadas y registro *Think-Aloud*; por tanto, los resultados se interpretan de manera descriptiva y exploratoria, sin pretensión de inferencia estadística poblacional.
- **Adaptación de escritorio.** El diseño UX/UI se desarrolló con enfoque *mobile-first*. La versión de escritorio es funcional y permite demostrar el prototipo, pero corresponde a una adaptación temporal del sistema móvil. Por limitaciones de tiempo no se diseñó una experiencia específica para la comodidad del usuario en PC, por lo que su ergonomía no debe interpretarse como un sistema *desktop* optimizado.
- **Cables, conexiones y electrónica detallada.** El prototipo no incluye la totalidad de cables, conectores, arneses ni elementos electrónicos internos del dron. Integrar esa capa habría requerido modelado adicional, revisión de rutas de conexión y desarrollo de interacciones específicas dentro de Unity para consultar o aislar dichas conexiones.

Por restricciones de tiempo, el alcance se concentró en la estructura principal, componentes representativos y herramientas de inspección visual.

- **Herramientas ocultas o no integradas.** Algunas capacidades permanecen en código, pero no forman parte de la UI final visible ni del flujo funcional documentado como estado final del producto.

Marco de Referencia

El diseño, desarrollo y evaluación del prototipo web 3D se fundamenta en un marco de referencia multidisciplinario. Este marco integra conocimientos de visualización técnica, interacción humano-computador, psicología cognitiva, computación gráfica, despliegue web con Unity y madurez de representaciones digitales de hardware complejo. El capítulo presenta primero el estado del arte de las soluciones disponibles para visualización 3D en navegador y de la evolución hacia *digital shadow/digital twin*; luego desarrolla el sustento teórico del proyecto y, finalmente, fija la terminología operativa utilizada en el resto del informe.

Estado del Arte

Esta sección examina bibliotecas nativas, motores de juego, plataformas *SaaS* y soluciones de *streaming* para identificar la herramienta más adecuada en términos de control del *pipeline*, capacidad de extensión y viabilidad de desarrollo dentro del alcance académico del proyecto.

Digital twins, digital shadows e interoperabilidad industrial

La literatura sobre *digital twins* coincide en que el valor de una representación digital no proviene únicamente del modelo 3D, sino de su capacidad para conectar entidad física, entidad virtual, estado, datos, procesos y frecuencia de sincronización. Kritzinger et al. (2018) proponen diferenciar *digital model*, *digital shadow* y *digital twin* según el flujo de datos entre el objeto físico y su representación digital.

Jones et al. (2020) amplían esa caracterización al incluir elementos como entidad física, entidad virtual, estado, metrología, procesos físicos y virtuales, conexión físico-virtual y *twinning rate*. En la misma línea, el Digital Twin Consortium (2020) define el gemelo digital como una representación virtual sincronizada con entidades y procesos reales, con frecuencia y fidelidad

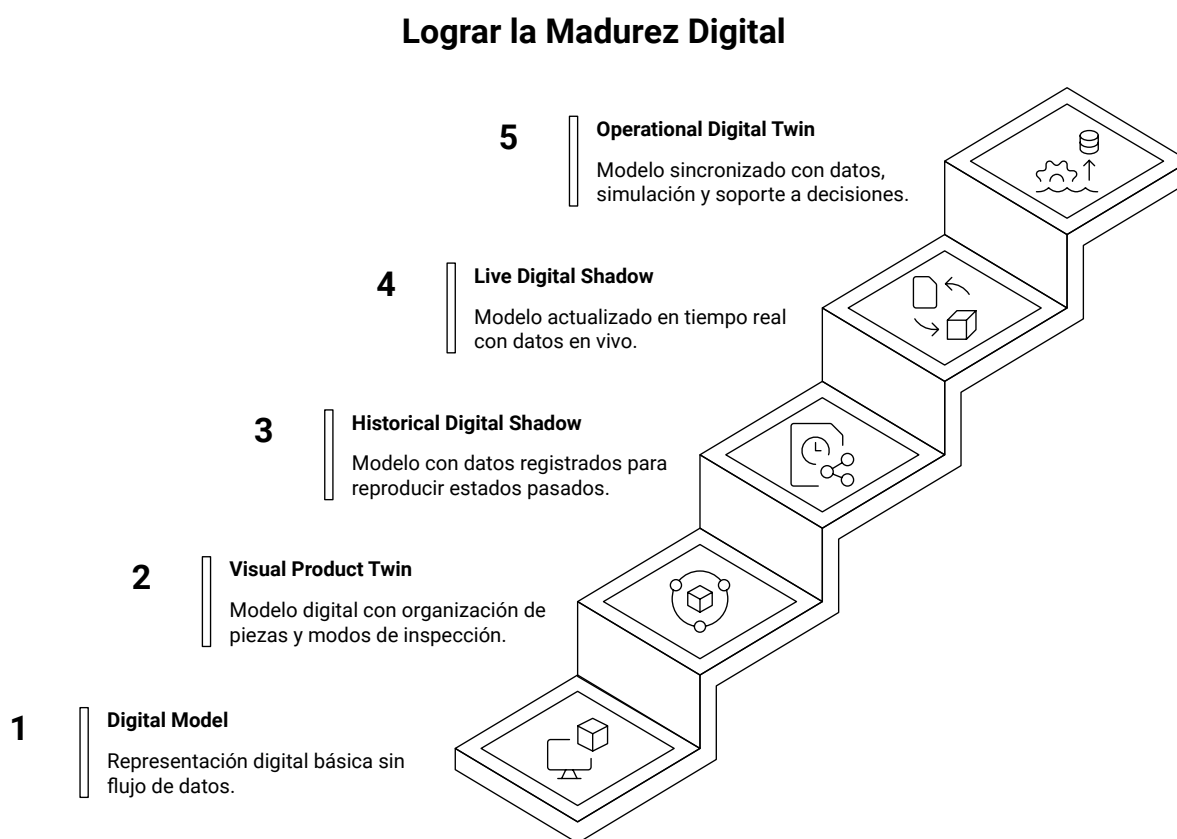
determinadas, orientada a apoyar resultados concretos.

Esta diferenciación es decisiva para el presente trabajo. Un visor 3D interactivo sin telemetría real, historial de estado ni sincronización automática no debe denominarse *digital twin* operacional. Su categoría más precisa es la de *digital model* enriquecido o *visual product twin*: una capa visual-semántica que permite inspeccionar y explicar el producto, y que puede servir como base para futuras integraciones de datos. La evolución hacia *digital shadow* requeriría incorporar datos históricos o en vivo del activo físico, mientras que un *digital twin* operacional exigiría además modelos calibrados, análisis, simulación o apoyo verificable a decisiones de mantenimiento, operación o ingeniería (Digital Twin Consortium, 2020; Fuller et al., 2020; Sharma et al., 2022; véase la Figura 4).

La interoperabilidad aparece como una barrera transversal. NIST identifica retos de integración cuando múltiples sistemas, proveedores, etapas del ciclo de vida y aplicaciones deben intercambiar información con significado preservado (David et al., 2024; Lin et al., 2023). En ese contexto, ISO 23247 es una familia de estándares para *digital twins* de manufactura que propone separar elementos observables, comunicación con dispositivos, entidad digital y aplicaciones de usuario. A su vez, el World Wide Web Consortium (W3C), organismo responsable de estándares web, plantea mediante Web of Things descripciones de propiedades, acciones y eventos para reducir la fragmentación de ecosistemas IoT; es decir, de dispositivos físicos conectados capaces de producir o consumir datos en red (International Organization for Standardization [ISO], 2021; W3C, 2023). Estas referencias sustentan la necesidad futura de un *twin manifest* por pieza, independiente de Unity, que conecte geometría, semántica, variables operativas, procedimientos y sistemas externos (véase la Figura 5).

Figura 4

Escalera de madurez, desde digital model hasta digital twin operacional.

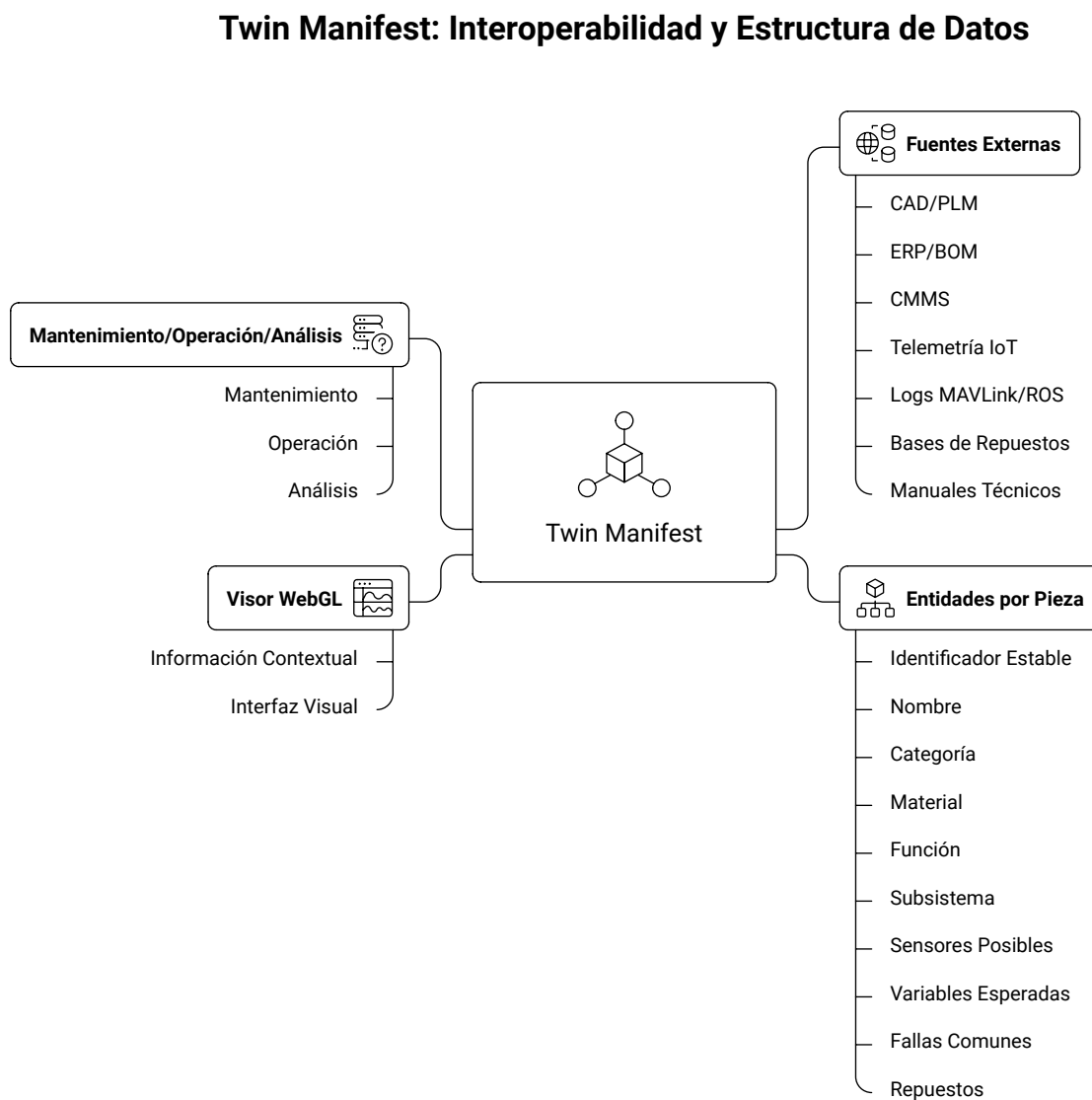


Nota. La build actual se ubica antes de las capas con telemetría histórica, datos en vivo y sincronización operacional.

Elaboración propia.

Figura 5

Interoperabilidad futura mediante twin manifest por componente.



Nota. El *twin manifest* se presenta como propuesta de evolución, no como módulo implementado en la versión evaluada. Elaboración propia.

Benchmarking de soluciones Web 3D

La visualización 3D en la web ha evolucionado de forma significativa desde la introducción de WebGL (Khronos Group, 2011). Yu et al. (2023) identifican a WebGL como la tecnología dominante para renderizado 3D en navegadores, con la aparición progresiva de nuevos *backends* como WebGPU. En este contexto, se revisaron las principales alternativas relevantes para el caso de estudio usando criterios explícitos de comparación: control del *pipeline*, posibilidad de lógica propia, integración de materiales y UI, necesidad de infraestructura externa, compatibilidad con despliegue web directo y esfuerzo de adopción dentro del alcance académico del proyecto (como se resume en la Tabla 1).

Three.js. Biblioteca ligera construida sobre WebGL y ampliamente adoptada por la comunidad de desarrollo web. Destaca por su huella inicial reducida y su flexibilidad para construir visualización 3D a medida; como contrapartida, exige ensamblar de forma más manual la importación, el manejo de escenas, la UI técnica y varias capas del *pipeline* que en este proyecto se resolvieron dentro de un entorno integrado (Cabello, 2024).

Babylon.js. Motor WebGL/WebGPU con soporte para escenas complejas, GUI, física y control amplio del *pipeline*. Si bien su documentación refleja una base madura para aplicaciones web 3D avanzadas, adoptarla habría implicado construir y mantener más componentes del flujo técnico fuera de un editor unificado, particularmente en importación CAD, perfilado y tooling auxiliar del visor (Babylon.js, s. f.).

Unity Web. Motor con compilación a web mediante la cadena $C\# \rightarrow IL2CPP \rightarrow C++ \rightarrow WebAssembly$. Su ecosistema integra editor visual, *profiler*, *pipeline* de materiales, herramientas para UI y soporte para extensión mediante código C# y shaders personalizados (Unity Tech-

nologies, 2024a, 2024b, 2024c). La documentación actual de Unity 6 contempla compatibilidad con navegadores compatibles de escritorio y algunos navegadores móviles, aunque el comportamiento efectivo depende del navegador, la versión, la memoria disponible y el nivel de optimización del proyecto (Unity Technologies, s. f.). Su principal costo técnico es una huella inicial de *build* mayor que la de bibliotecas puramente web.

Unreal Engine Pixel Streaming. Solución de *streaming* remoto que depende de infraestructura GPU, servidores de señalización y render del lado del servidor. Resulta potente para experiencias de alto costo gráfico, pero para este proyecto añadía una capa de infraestructura, red y latencia que se alejaba del objetivo de un *build* web de acceso directo en navegador (Epic Games, s. f.-a, s. f.-b).

Spline. Herramienta *web-first* orientada a diseño iterativo, escenas embebidas y publicación rápida. Sus exportaciones, visores y modos de publicación son útiles para experiencias visuales ligeras; no obstante, su modelo de trabajo está más centrado en escenas y *embeds* que en un sistema técnico con saneamiento de escena, lógica runtime propia y tooling especializado como el requerido aquí (Spline, s. f.-a, s. f.-b, s. f.-c).

Marmoset Viewer. Visor WebGL especializado en presentación PBR de alta calidad. Su documentación oficial muestra un conjunto de funciones orientadas a exhibición visual del modelo y un subconjunto específico de características soportadas; para este proyecto, ese enfoque resultaba insuficiente frente a necesidades como lógica contextual, instrumentación de tareas, clipping global y acoplamiento profundo con UI y datos del sistema (Marmoset, s. f.).

Sketchfab. Plataforma de *hosting* y visualización 3D con publicación rápida y una Viewer API capaz de controlar cámara, materiales, anotaciones y eventos. Aun así, su integración se organiza alrededor de un visor embebido y no de un *runtime* integral del proyecto, por lo que

ofrecía menos alineación con la necesidad de una arquitectura propia, un *bottom sheet* técnico y herramientas analíticas específicas del artefacto (Sketchfab, s. f.).

PlayCanvas. Plataforma de desarrollo web con editor colaborativo, *runtime* ligero, scripting integrado y fuerte orientación web nativa. Es una alternativa seria para experiencias 3D en navegador; sin embargo, para este caso seguía implicando adoptar otro entorno, reconstituir el *pipeline* de importación y rehacer parte del tooling técnico ya consolidado en Unity para inspección, profiling y organización del proyecto (PlayCanvas, s. f.-a, s. f.-b, s. f.-c).

Tabla 1

Comparativa cualitativa de soluciones Web 3D

Tecnología	Huella inicial	Control del pipeline	PBR	Interactividad	Infraestructura externa	Pertinencia
Three.js	Baja	Alto, pero manual	Manual/extendible	Total	No	Alta para desarrollo a medida
Babylon.js	Media	Alto	Integrado	Total	No	Media/alta para un stack web propio
Unity Web	Alta	Alto e integrado	URP	Total	No	Alta para este caso de estudio
UE Pixel Streaming	Media/Alta	Alto	Muy alto	Alta con latencia	Sí	Baja/media para acceso web directo
Spline	Baja	Bajo	Básico	Limitada	No	Baja/media para este caso
Marmoset Viewer	Baja/Media	Bajo	Alto	Baja/Media	No	Baja/media para este caso
Sketchfab	Media	Medio	Bueno	Media	Parcial	Media para publicación guiada
PlayCanvas	Baja	Alto	Integrado	Total	No	Media/alta para experiencias web nativas

Nota. Comparación cualitativa basada en documentación oficial, literatura revisada y requerimientos del proyecto; no corresponde a un benchmark experimental uniforme. Los juicios de pertinencia se refieren a este caso de estudio. Elaboración propia.

Brecha identificada y selección de Unity Web

La evaluación técnica identificó limitaciones críticas en tres grupos de soluciones:

- **Pixel Streaming.** Dependencia de infraestructura de render remoto, red y configuración externa poco compatibles con un prototipo académico de acceso directo en nave-

gador.

- **Spline, Marmoset y Sketchfab.** Menor alineación con una arquitectura propia que exigía lógica de inspección técnica, flujo de detalle, modos analíticos y control fino del sistema.
- **Three.js, Babylon.js y PlayCanvas.** Mayor esfuerzo de adopción para reconstruir un *pipeline* integral de importación, materiales, controles, clipping global, modos de visualización técnica y herramientas auxiliares de auditoría dentro del tiempo académico disponible.

Unity Web fue seleccionado no por ofrecer la huella inicial más baja, sino por el equilibrio entre cuatro factores:

- **Integración del tooling.** Permite concentrar importación, materiales, UI, perfilado y build en un mismo entorno.
- **Herramientas visuales y de editor.** Facilita el trabajo conjunto entre programación, diseño y arte técnico.
- **Arquitectura tipada y extensible.** La cadena $C\# \rightarrow IL2CPP \rightarrow C++ \rightarrow WASM$ ofrece una base estructurada para la lógica interactiva del prototipo.
- **Soporte para modos analíticos.** Reduce el riesgo técnico al implementar corte transversal, visualización térmica, vista *X-Ray* y otras herramientas especializadas dentro del tiempo académico disponible.

Marco Teórico

El marco teórico integra cuatro pilares: madurez de representaciones digitales de hardware complejo, teoría de la carga cognitiva e interacción, computación gráfica en tiempo real y

fundamentos matemáticos de optimización y sombreado. Estos fundamentos sustentan la arquitectura, las decisiones de implementación, la estrategia de evaluación del prototipo y el roadmap de evolución hacia capacidades de *digital shadow*.

Madurez de integración digital en hardware complejo

La clasificación de Kritzinger et al. (2018) permite ordenar el proyecto como una trayectoria incremental. En la primera etapa, un *digital model* describe el sistema sin flujo automático de datos con el activo físico. En la segunda, un *digital shadow* recibe datos del activo y actualiza su estado digital. En una etapa más avanzada, un *digital twin* operacional combina sincronización, conocimiento de dominio, datos históricos o actuales y capacidades de análisis o simulación para apoyar decisiones (Digital Twin Consortium, 2020; Jones et al., 2020).

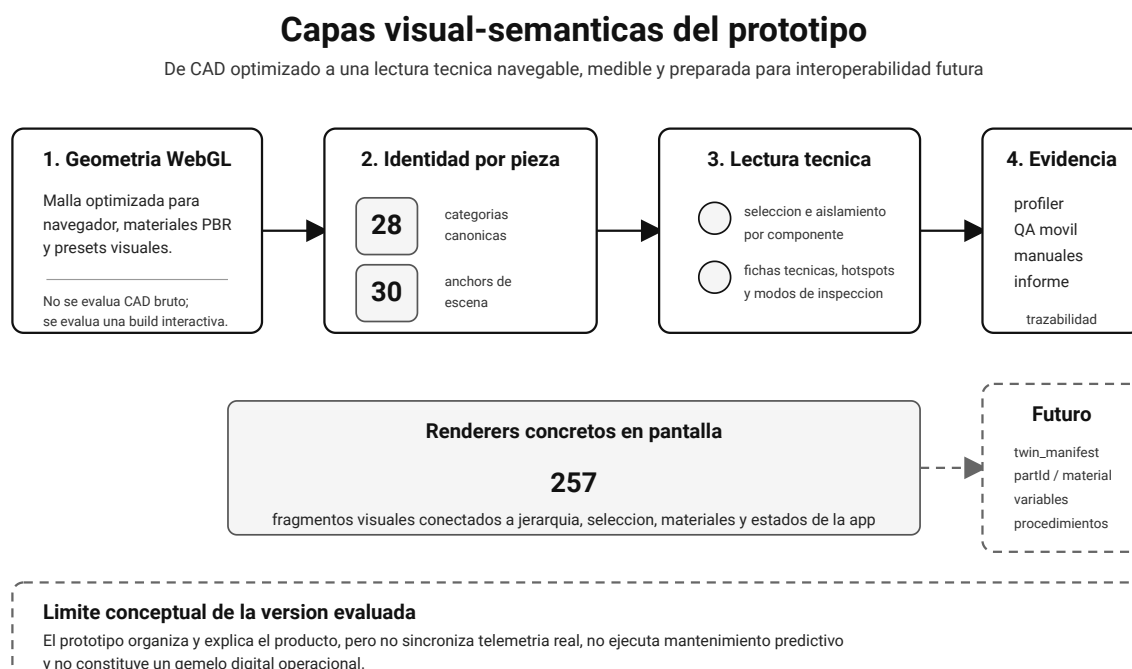
Desde este marco, el prototipo desarrollado se ubica en la primera etapa, con una particularidad relevante: no es un CAD bruto ni una escena 3D decorativa, sino un modelo visual-semántico con taxonomía por piezas, selección, aislamiento, fichas técnicas, modos analíticos, profiler interno y documentación asociada. Esa capa resuelve necesidades de comunicación técnica y prepara el terreno para un *twin manifest*, pero todavía no incorpora telemetría ni servicios de mantenimiento. Esta delimitación evita una inflación conceptual del término *digital twin* y permite defender el aporte actual como base técnica e investigativa para fases posteriores (véase la Figura 6).

Teoría de la Carga Cognitiva (Sweller, 1988; Sweller et al., 2019)

La Teoría de la Carga Cognitiva plantea que la memoria de trabajo tiene capacidad limitada y que el diseño de materiales e interfaces debe regular la demanda que impone sobre ella. Según Cowan (2001), esta capacidad operativa ronda unos pocos elementos simultáneos, por lo que la forma de presentar información técnica incide directamente en la comprensión y en la fati-

Figura 6

Capas del prototipo como modelo visual-semántico de producto.



Nota. La lectura visual-semántica del prototipo se apoya en piezas canónicas, *anchors* de escena y *renderers* concretos; la interoperabilidad mediante *twin manifest* queda delimitada como evolución futura. Elaboración propia.

ga del usuario.

Sweller, van Merriënboer y Paas (2019) distinguen tres tipos de carga:

Carga intrínseca. Corresponde a la complejidad inherente del contenido. En este proyecto es elevada porque el dron integra componentes estructurales, energéticos, electrónicos y mecánicos con relaciones espaciales y funcionales entre sí.

Carga extrínseca. Es la carga innecesaria generada por una presentación deficiente. En documentación 2D estática, el usuario debe imaginar relaciones tridimensionales no visibles, lo cual incrementa la necesidad de rotación mental y consume recursos cognitivos que no están dirigidos al objetivo técnico principal.

Carga germana. Es el esfuerzo que sí contribuye a construir esquemas mentales útiles. En una interfaz 3D bien diseñada, los recursos liberados por la reducción de carga extrínseca pueden redirigirse a comprender mejor la función, ubicación y relación entre componentes del ensamblaje.

En el contexto del prototipo, la interfaz 3D orbital, la selección directa sobre el modelo y la consulta contextual de piezas buscan reducir la carga extrínseca y favorecer una carga germana orientada a la comprensión espacial y funcional del ensamblaje. No obstante, cuando la evaluación usa NASA-TLX, la variable observada es carga de trabajo percibida y no una medición directa de las categorías intrínseca, extrínseca y germana de la teoría (véase la Figura 7).

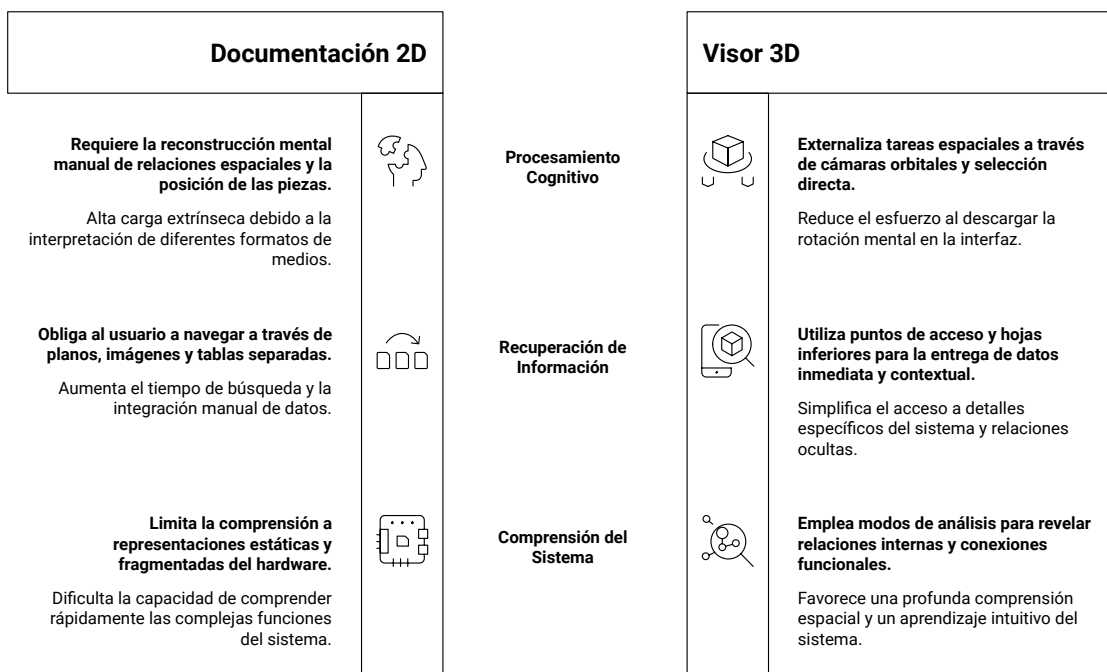
Navegación espacial e interacción 3D

Bowman et al. (2004) definen la interacción 3D como aquella en la cual las tareas del usuario se ejecutan directamente en un contexto espacial tridimensional. Esto es relevante porque el valor del prototipo depende de que el usuario pueda explorar el ensamblaje sin perder contexto global mientras inspecciona detalles locales.

Figura 7

Relación entre documentación 2D, carga cognitiva y apoyo visual 3D.

¿Qué método de visualización minimiza la carga cognitiva para hardware complejo?



Nota. La figura no afirma que el visor elimine la complejidad intrínseca del dron; muestra cómo puede reducir carga extrínseca asociada a búsqueda, rotación mental e integración manual de información. Elaboración propia.

Los controles orbitales siguen principios de navegación espacial orientados a prevenir desorientación:

- rotación alrededor de un foco de interés estable;
- *pan* y *zoom* como operaciones diferenciadas de la rotación;
- recentrado del interés cuando se selecciona una pieza o se invoca una herramienta de inspección;
- continuidad de movimiento suficiente para evitar saltos bruscos de interpretación espacial.

Cognición distribuida (Hutchins, 1995). La interfaz no solo presenta información: también funciona como artefacto cognitivo. La cámara orbital externaliza parte de la rotación mental, los *hotspots* y fichas de pieza operan como memoria externa y los modos analíticos convierten procesos de inspección complejos en operaciones visuales guiadas.

Heurísticas de usabilidad (Nielsen, 1994)

Las heurísticas de Nielsen orientan la construcción de la interfaz:

1. **Visibilidad del estado del sistema.** La aplicación debe comunicar con claridad qué pieza está seleccionada, qué modo visual está activo y cuándo una herramienta modifica la escena.
2. **Correspondencia con el mundo real.** Los nombres de componentes, la organización por categorías y la lógica de inspección deben alinearse con la terminología técnica del dron.
3. **Control y libertad del usuario.** La navegación, la salida de paneles y el reinicio de estado deben ser previsibles y reversibles.
4. **Reconocimiento antes que memoria.** La interfaz debe reducir la necesidad de recor-

dar nombres, ubicaciones o secuencias de interacción, apoyándose en ayudas visuales y estados explícitos.

Rotación mental y visualización espacial

Hegarty y Waller (2004) muestran que la interpretación de estructuras tridimensionales depende de habilidades espaciales que no se distribuyen homogéneamente entre usuarios. Bartlett y Dorribo Camba (2023) refuerzan que los modelos 3D interactivos pueden disminuir esta barrera al facilitar interpretación gráfica y comprensión del ensamblaje. En consecuencia, el visor no se justifica solo por atractivo visual, sino como apoyo cognitivo para usuarios con distintos niveles de habilidad espacial.

Percepción visual y organización de la información

Los principios de Gestalt (Wertheimer, 1923; Köhler, 1929) ayudan a estructurar la escena y la interfaz:

- **Proximidad.** Elementos cercanos se perciben como pertenecientes al mismo subconjunto.
- **Similitud.** El color o tratamiento visual de *hotspots* y categorías facilita asociación funcional.
- **Figura-fondo.** El modelo principal debe destacar sobre el fondo para no competir con el contenido técnico.

Ware (2012) y Miller (1956) sustentan el uso de categorías cromáticas distinguibles, evitando esquemas que dificulten accesibilidad o saturen la memoria inmediata del usuario.

Fundamentos matemáticos de optimización gráfica

El rendimiento de una aplicación WebGL no depende de una sola variable, sino del equilibrio entre complejidad geométrica, número de lotes renderizados, costo de sombreado, ancho de

banda de memoria y tiempo total por cuadro. Por ello, las ecuaciones de esta sección se presentan como modelos operativos para análisis y control de presupuesto, no como leyes universales aisladas del contexto de implementación (véase la Figura 8).

Figura 8

Balance entre fidelidad visual, legibilidad técnica y rendimiento WebGL.



Nota. La figura resume el criterio de *Technical Art* del proyecto: preservar reconocimiento visual y lectura técnica sin romper el presupuesto de ejecución en navegador. Elaboración propia.

Presupuesto poligonal. Una primera restricción de complejidad geométrica puede expresarse como:

$$P_{\text{total}} = \sum_{i=1}^n P_i$$

donde P_i representa los triángulos aportados por cada componente del modelo. En términos de planeación, el proyecto fija un presupuesto objetivo para mantener la escena dentro de un rango compatible con WebGL. Este presupuesto no es una ley universal del motor, sino un criterio de control para el proceso de optimización.

Densidad de texel. La densidad de texel busca que la nitidez percibida de las texturas sea coherente entre componentes. Una forma operativa de expresarla es:

$$\rho = \frac{R_{\text{texture}}}{L_{\text{mundo}}}$$

donde R_{texture} es la resolución efectiva aplicada sobre un tramo de referencia y L_{mundo} es la longitud real correspondiente en el modelo. Si ρ cambia bruscamente entre piezas, algunas se perciben borrosas y otras resultan innecesariamente costosas en memoria.

Costo de dibujo y agrupación de mallas. El costo de renderizado no depende solo del número de triángulos, sino también del número de lotes o *draw calls*. Sin agrupación, una aproximación simple es:

$$D_{\text{base}} \approx \sum_{k=1}^m n_k$$

donde n_k representa la cantidad de objetos que requieren una combinación distinta de malla, material o pasada. Si existen condiciones de *batching* o *instancing* compatibles, varias instancias

pueden reducirse a menos lotes efectivos:

$$D_{\text{efectivo}} \approx \sum_{k=1}^m b_k, \quad 1 \leq b_k \leq n_k$$

Por tanto, la reducción de *draw calls* no debe modelarse como una división exacta universal, sino como una función de compatibilidad entre materiales, mallas, pasadas de render y estado del *pipeline*.

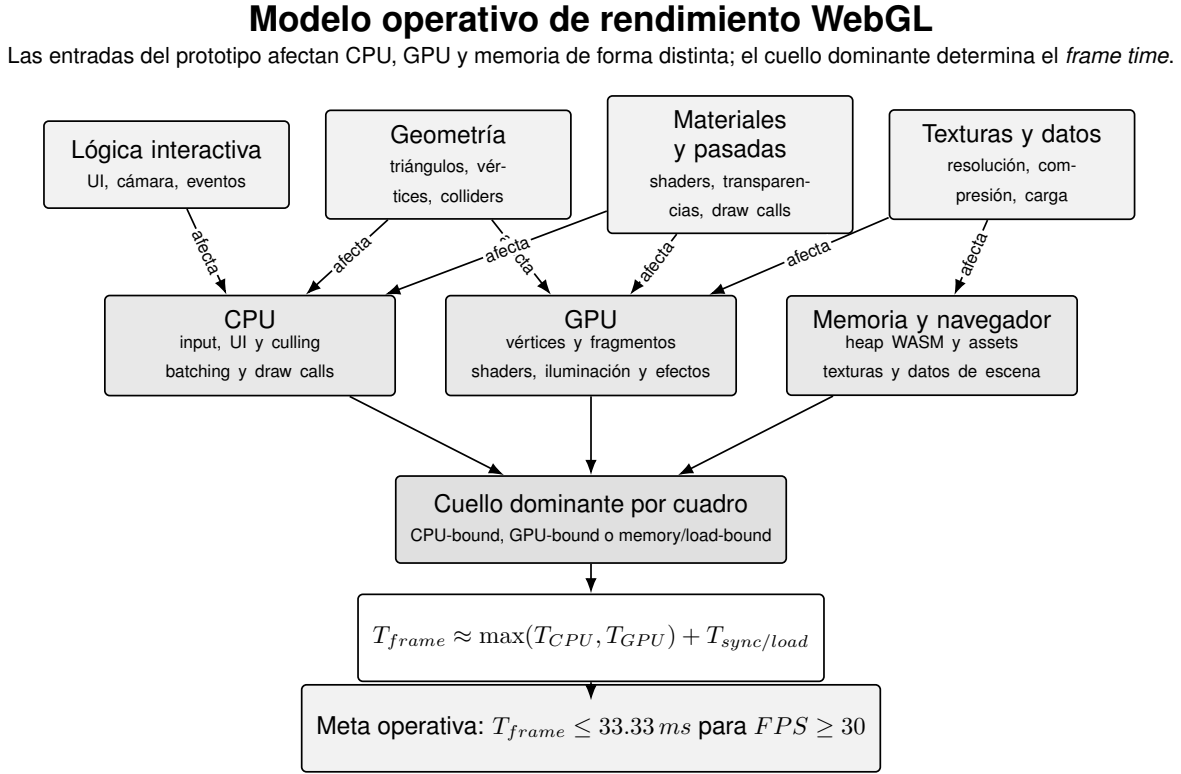
Frame time y objetivo de FPS.. En una aplicación interactiva, el tiempo por cuadro determina la fluidez percibida. Una aproximación útil para perfilado es:

$$T_{\text{frame}} \approx \text{máx}(T_{\text{CPU}}, T_{\text{GPU}})$$

con un posible costo adicional por sincronización. Para sostener 30 FPS, se requiere aproximadamente:

$$T_{\text{frame}} \leq 33,33 \text{ ms}$$

Esto traduce el rendimiento a una restricción directamente medible en navegador y en herramientas de perfilado (véase la Figura 9).

Figura 9*Modelo operativo de rendimiento WebGL.*

Nota. El tiempo de cuadro efectivo se interpreta como resultado del cuello dominante entre CPU, GPU, memoria/carga y sincronización, no como efecto de una sola técnica de optimización. Elaboración propia.

Modelo matemático de renderizado físicamente basado (PBR)

El renderizado físicamente basado parte de la *rendering equation*, formulada por Kajiya (1986) y adoptada como fundamento general de los modelos BRDF utilizados en renderizado en tiempo real (Akenine-Möller et al., 2018):

$$L_o(\omega_o) = \int_{\Omega} f_r(\omega_i, \omega_o) L_i(\omega_i) (\mathbf{n} \cdot \omega_i) d\omega_i$$

donde L_o es la radiancia saliente en la dirección de observación ω_o , L_i es la radiancia incidente desde la dirección ω_i , $\mathbf{n}$ es la normal de la superficie y f_r es la BRDF del material. Esta ecuación expresa que el color visible depende de cuánta luz llega a la superficie, cómo la superficie la redistribuye y qué parte de esa redistribución coincide con la dirección de la cámara.

BRDF especular de Cook-Torrance. La componente especular puede modelarse como:

$$f_{\text{spec}} = \frac{D(\mathbf{n}, \mathbf{h}) F(\mathbf{v}, \mathbf{h}) G(\mathbf{n}, \mathbf{l}, \mathbf{v})}{4(\mathbf{n} \cdot \mathbf{l})(\mathbf{n} \cdot \mathbf{v})}$$

donde $\mathbf{l}$ es la dirección de la luz, $\mathbf{v}$ la dirección de vista y $\mathbf{h}$ el vector mitad entre ambas. Cada término cumple una función distinta:

- D describe la distribución estadística de microfacetas orientadas para reflejar luz hacia la cámara;
- F modela el efecto Fresnel, es decir, el incremento de reflectancia cuando la observación se acerca al ángulo rasante;
- G atenúa la respuesta especular por auto-sombreado y enmascaramiento geométrico entre microfacetas.

Aproximación de Fresnel de Schlick. Para reducir costo computacional, el término de Fresnel suele aproximarse mediante la expresión de Schlick (1994):

$$F = F_0 + (1 - F_0)(1 - \cos \theta)^5$$

donde F_0 es la reflectancia a incidencia normal y θ es el ángulo entre la dirección de vista y el vector mitad. Esta aproximación permite obtener un comportamiento visualmente creíble con un

costo bajo en tiempo real.

Distribución GGX. La distribución de microfacetas adoptada comúnmente en PBR en tiempo real es GGX (Walter et al., 2007; Karis, 2013):

$$D_{\text{GGX}} = \frac{\alpha^2}{\pi ((\mathbf{n} \cdot \mathbf{h})^2 (\alpha^2 - 1) + 1)^2}$$

donde α controla la amplitud del lóbulo especular y suele derivarse de la rugosidad como $\alpha = \text{roughness}^2$. Valores bajos producen reflejos concentrados y valores altos reflejos anchos y difusos.

Conservación de energía. En un flujo *metallic-roughness*, la energía reflejada debe mantenerse acotada. De manera simplificada:

$$k_d + k_s \leq 1$$

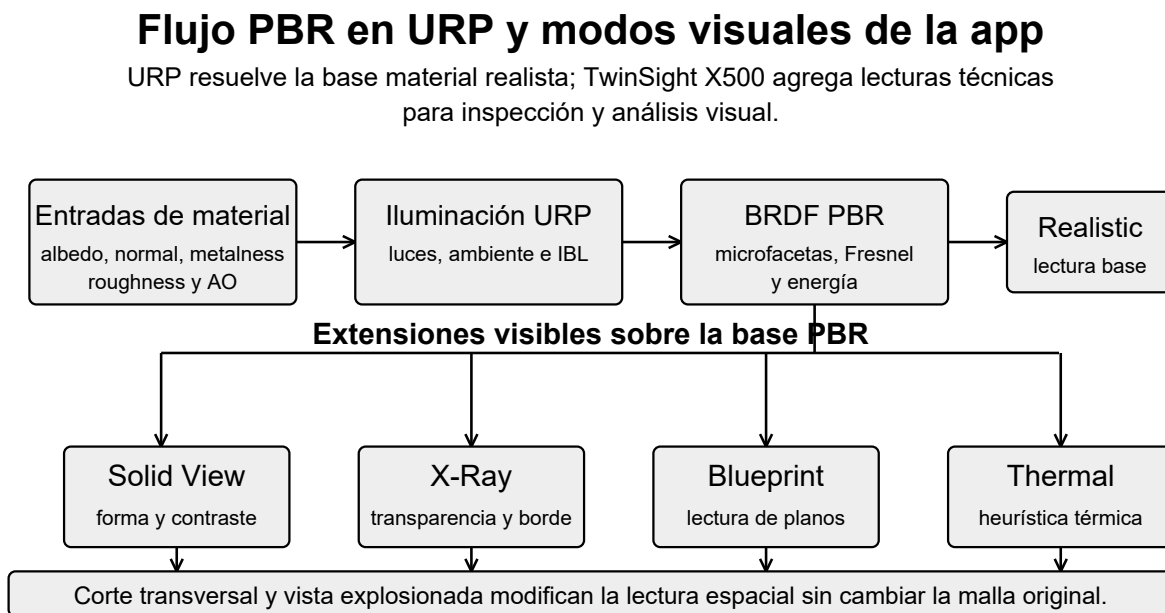
donde k_d representa la fracción difusa y k_s la especular. En materiales metálicos, la componente difusa tiende a desaparecer y el color especular toma protagonismo; en dieléctricos ocurre lo contrario.

Relación con la app real. En la aplicación final, el modo *Realistic* no reimplementa manualmente toda la BRDF microfacetada desde cero. El proyecto delega el cálculo PBR base al *pipeline* de Unity URP, mientras que los shaders propios añaden lógica específica del sistema, como planos de corte globales y modos de inspección. Por ello, las ecuaciones anteriores se presentan como fundamento matemático del modelo de iluminación empleado por URP y no como afirmación de un *solver* BRDF completamente escrito desde cero dentro del proyecto.

Iluminación basada en imágenes (IBL). La iluminación ambiental en PBR se apoya en mapas de entorno precalculados. Conceptualmente, la integral ambiental se aproxima mediante dos componentes: un mapa de irradiancia para la parte difusa y un mapa de entorno prefiltrado para la parte especular. Esto permite aproximar reflejos y contribución ambiental con costo constante por píxel, sin necesidad de *ray tracing* completo en tiempo real (véase la Figura 10).

Figura 10

Lectura visual del flujo PBR usado por URP en la app.



Nota. El diagrama distingue la base PBR delegada a URP de las extensiones propias de inspección técnica, como corte, X-Ray y lectura térmica heurística. Elaboración propia.

Matemática aplicada a shaders de inspección y análisis

Plano de corte global. El corte transversal de la app puede expresarse con la ecuación de un plano:

$$\Pi = (n_x, n_y, n_z, d), \quad d = -\mathbf{n} \cdot \mathbf{p}$$

donde $\mathbf{n}$ es la normal del plano y $\mathbf{p}$ un punto perteneciente a dicho plano. Para cualquier punto del espacio $\mathbf{x}$, la distancia algebraica al plano se evalúa como:

$$\mathbf{n} \cdot \mathbf{x} + d$$

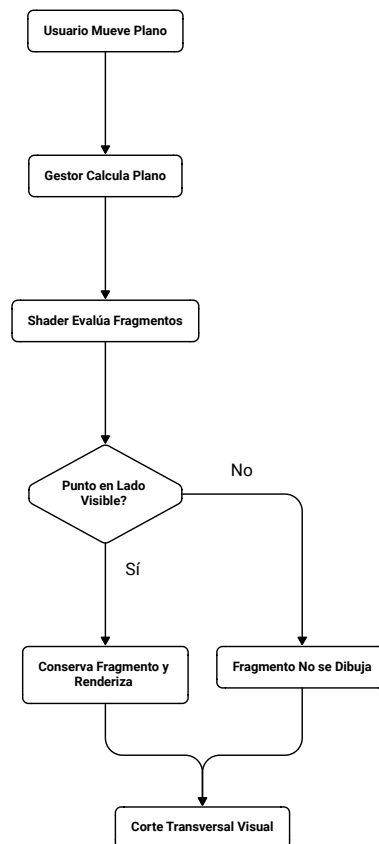
Si este valor es negativo, el fragmento queda del lado descartado del plano y no se renderiza:

$$\mathbf{n} \cdot \mathbf{x} + d < 0 \Rightarrow \text{fragmento descartado}$$

Esta formulación coincide con el mecanismo usado por los shaders del sistema cuando reciben los planos globales calculados por el gestor de corte (véase la Figura 11).

Figura 11

Proceso de corte transversal aplicado por shader.

Proceso de Corte Transversal de Aplicación

Nota. El esquema resume el paso desde el movimiento del plano hasta la evaluación fragmento a fragmento; no representa una operación destructiva sobre la malla. Elaboración propia.

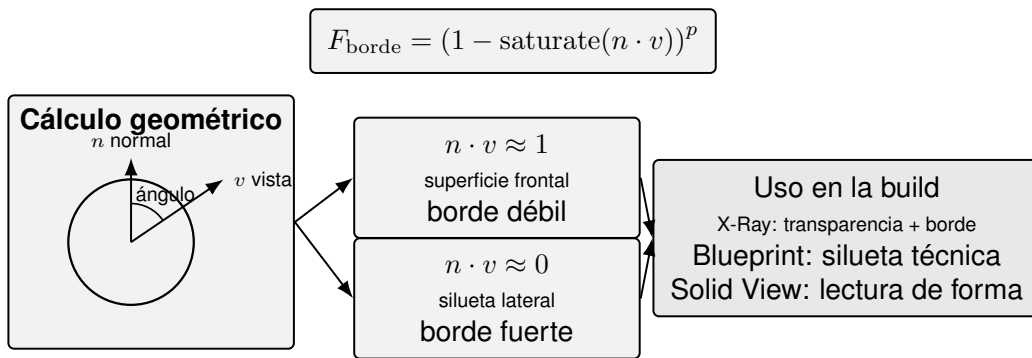
Fresnel de borde para modos analíticos. Modos como *X-Ray* o *Blueprint* usan una intensidad de borde basada en la relación entre normal y vista:

$$F_{\text{borde}} = (1 - \text{saturnate}(\mathbf{n} \cdot \mathbf{v}))^p$$

donde p controla la dureza del borde. Cuando la normal es casi perpendicular a la vista, la intensidad aumenta y se resaltan siluetas, contornos o regiones parcialmente transparentes (véase la Figura 12).

Figura 12

Intensidad Fresnel y contornos técnicos en sombreado de borde.



El contorno es una regla de sombreado para lectura visual; no modifica la geometría ni representa una medición física.

Nota. Representación del cálculo de Fresnel de borde basado en el ángulo entre la normal de superficie y la vista para resaltar siluetas. Elaboración propia.

Mapeo visual del modo térmico. La vista térmica del prototipo no corresponde a un *solver* FEA completo, sino a una visualización híbrida inspirada en un estado térmico simplifica-

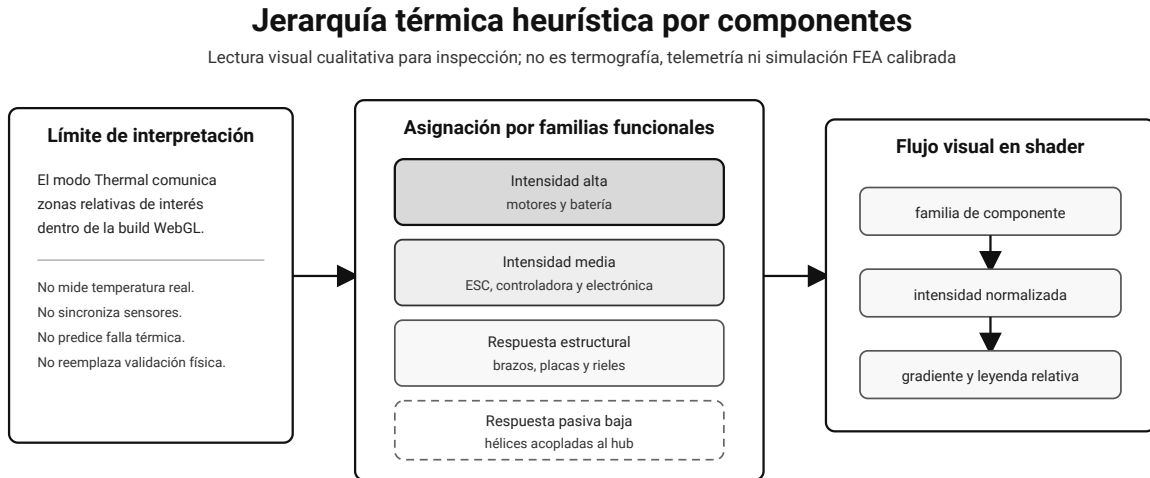
do. A nivel visual, la temperatura normalizada puede expresarse como:

$$T_{\text{vis}} = \text{saturate} (T_{\text{base}} s(\mathbf{x}) - \phi_{\text{edge}}(\mathbf{x}, \mathbf{v}) + \eta(\mathbf{x}, t) + \delta_{\text{textura}})$$

donde T_{base} proviene del subsistema térmico, $s(\mathbf{x})$ representa el factor espacial radial o axial, $\phi_{\text{edge}}(\mathbf{x}, \mathbf{v})$ modela el enfriamiento visual de borde, $\eta(\mathbf{x}, t)$ introduce variación procedural suave y δ_{textura} incorpora microvariaciones visuales de la textura base. Esta ecuación describe una capa de visualización, no una simulación termo-mecánica completa (véase la Figura 13).

Figura 13

Jerarquía térmica heurística por componentes.



La figura separa fuentes activas, piezas estructurales y piezas pasivas para evitar que el color sugiera una temperatura física medida.

Nota. Clasificación de disipación térmica heurística entre componentes activos (fuentes de calor primarias) y pasivos del dron Holybro X500 V2. Elaboración propia.

Marco Conceptual

Los siguientes conceptos técnicos fundamentan el proyecto. Algunos corresponden a componentes efectivamente utilizados en la aplicación final y otros a estrategias de optimización evaluadas durante el diseño del *pipeline*; no todos están expuestos como funcionalidad visible en la build final.

Albedo. Mapa de textura que define el color base de un material sin incorporar iluminación. Es una de las entradas principales del flujo PBR.

Asset Bundle. Mecanismo de empaquetado de Unity para distribuir activos en bloques independientes. En este proyecto se consideró como alternativa de despliegue, pero no constituye el eje central documentado para la versión final de la aplicación.

Baking. Proceso de precalcular información compleja, como detalle geométrico o iluminación, y almacenarla en texturas para reducir costo computacional en tiempo real.

BRDF (Bidirectional Reflectance Distribution Function). Función que describe cómo una superficie refleja luz según la dirección de incidencia y de observación. Es el fundamento matemático del renderizado físicamente basado.

C#. Lenguaje de programación principal usado en Unity para implementar la lógica interactiva del prototipo, incluyendo selección, UI, managers de escena y herramientas de análisis visual.

C++. Lenguaje de bajo nivel usado como representación intermedia dentro de la cadena IL2CPP de Unity antes de compilar el target web a WebAssembly.

Draw Call. Comando emitido por la CPU para que la GPU renderice un lote de geometría con un material y una pasada concretos. Su reducción es clave para el rendimiento.

Fragment Shader (Pixel Shader). Programa ejecutado por la GPU para cada fragmento visible, encargado de calcular el color final.

Frame Time. Tiempo efectivo requerido para producir un cuadro. En una aproximación operativa de perfilado suele modelarse como $T_{\text{frame}} \approx \max(T_{\text{CPU}}, T_{\text{GPU}})$, más el posible costo de sincronización entre ambos.

GPU Instancing. Técnica que permite renderizar varias instancias compatibles de una misma malla y material en menos lotes efectivos. La reducción de *draw calls* depende de las condiciones de compatibilidad y no de una división exacta universal.

Hardware complejo. Categoría operativa utilizada en este informe para referirse a sistemas electromecánicos multicomponente con integración estructural, energética, electrónica y de control, alta densidad de relaciones espaciales entre piezas y necesidad de inspección visual detallada del ensamblaje.

Digital model. Representación digital de un objeto o sistema físico en la que los cambios no se transfieren automáticamente entre el activo real y su contraparte digital. En este proyecto, el prototipo WebGL se aproxima a esta categoría enriquecida por una capa visual-semántica.

Digital shadow. Representación digital que recibe datos del activo físico de manera automática o semiautomática, de forma que el estado real actualiza el estado digital. Requiere telemetría histórica o en vivo y no forma parte del alcance implementado en la build final.

Digital twin. Representación virtual sincronizada con entidades o procesos reales a una frecuencia y fidelidad determinadas, capaz de apoyar comprensión, análisis, simulación o decisiones operacionales. En este informe se reserva el término para trabajo futuro con datos reales, modelos calibrados e integración operacional.

Visual product twin o technical product twin. Categoría operativa propuesta para describir

la capa actual del proyecto: un modelo web 3D interactivo, semántico y optimizado de un producto físico, orientado a comprensión, inspección, comunicación técnica y preparación de datos.

Twin manifest. Contrato de datos por componente que conectaría identificador de pieza, geometría, subsistema, material, función, sensores posibles, variables de estado, fallas comunes, repuestos, procedimientos y documentos asociados. Se plantea como trabajo futuro para habilitar interoperabilidad.

Digital thread. Continuidad de datos que conecta diseño, manufactura, ensamblaje, operación, mantenimiento y retiro del producto. Un *digital twin* confiable requiere trazabilidad entre estas etapas para preservar significado técnico.

Interoperabilidad. Capacidad de que modelos, aplicaciones, dispositivos y sistemas de distintos proveedores o etapas del ciclo de vida intercambien datos útiles con significado preservado. Es una condición central para escalar el visor hacia *digital shadow* o *digital twin*.

IL2CPP (Intermediate Language to C++). Backend de compilación de Unity que transforma el código intermedio (IL) de .NET generado a partir de C# en C++. En el target web, ese código C++ se compila posteriormente a WebAssembly mediante el *toolchain* correspondiente del build.

IoT (Internet of Things). Conjunto de dispositivos físicos conectados que pueden registrar, transmitir o recibir datos mediante redes. En el contexto del proyecto, IoT se entiende como una posible capa futura de sensores y eventos operativos, no como una capacidad implementada en la build actual.

ISO 23247. Familia de estándares orientada a un marco de gemelos digitales para manufactura. Su valor para este informe está en proponer una separación por capas entre elementos observables, comunicación, entidad digital y aplicaciones de usuario.

LOD (Level of Detail). Estrategia de optimización que intercambia versiones de diferente complejidad geométrica según distancia o relevancia visual. Se considera una técnica de referencia del *pipeline*, aunque su adopción final depende del contenido realmente exportado y configurado.

Metallic Map. Textura que define qué regiones del material deben tratarse como metálicas frente a dieléctricas, afectando la reflectancia especular base.

MVP (Producto Mínimo Viable). Versión de un producto con las funcionalidades suficientes para poner a prueba supuestos esenciales con usuarios reales y orientar iteraciones posteriores.

Normal Map. Textura que almacena variaciones de normales para simular detalle superficial sin aumentar el conteo de triángulos.

Occlusion Culling. Técnica que evita dibujar objetos no visibles porque quedan ocultos desde la perspectiva de la cámara. En este proyecto se asume como principio general de optimización, no como afirmación automática de uso en todos los escenarios de la build final.

PBR (Physically-Based Rendering). Modelo de iluminación que aproxima la interacción física entre luz y materiales mediante conservación de energía y teoría de microfacetas.

Retopología. Proceso de reconstruir la topología de una malla para optimizar su estructura geométrica y su viabilidad en tiempo real.

Roughness Map. Textura que controla la rugosidad microscópica de la superficie. Valores bajos producen reflejos nítidos; valores altos generan reflejos difusos.

Shader. Programa ejecutado por la GPU que participa en el cálculo visual de vértices o fragmentos.

Specular. Componente de reflexión dependiente del ángulo de observación y de la micro-

geometría del material.

Texel Density. Relación entre la resolución efectiva de la textura y la longitud real de superficie que representa. Puede expresarse como $\rho = R_{\text{texture}}/L_{\text{mundo}}$ en píxeles por centímetro cuando se toma una dirección o tramo de referencia.

UV Unwrapping. Proceso de desplegar una superficie 3D en coordenadas 2D para aplicar texturas con la menor distorsión posible.

Vertex Shader. Programa ejecutado por la GPU para cada vértice de la malla, responsable de transformar posiciones entre espacios geométricos.

WebAssembly (WASM). Formato binario ejecutable en navegadores que permite desplegar lógica compilada dentro de un modelo de memoria lineal, utilizado por Unity para ejecutar en web el código generado a partir de C#.

WebGL. API gráfica del navegador basada en OpenGL ES para renderizado 2D y 3D acelerado por hardware sin complementos externos.

W3C (World Wide Web Consortium). Organización internacional que desarrolla estándares para la Web. En este informe se cita por Web of Things, una línea de trabajo orientada a describir propiedades, acciones y eventos de dispositivos conectados para reducir fragmentación entre ecosistemas IoT.

Marco Metodológico

Este capítulo describe el diseño metodológico adoptado para el desarrollo y la evaluación del prototipo web del Holybro X500 V2. La investigación se formuló como aplicada, con enfoque mixto y predominio cualitativo-formativo. El trabajo articuló diseño, construcción y evaluación de un artefacto digital, por lo que adoptó *Design Science Research* (DSR) como marco principal, complementado con prototipado iterativo, observación de uso, cuestionarios estandarizados y medición de KPIs técnicos. La identificación del problema no se restringió al caso del dron, sino que partió de una necesidad industrial más amplia: la visualización, documentación y preparación semántica de hardware complejo.

Tipo de investigación y enfoque

- **Tipo de investigación:** aplicada, porque busca resolver una necesidad concreta de visualización técnica en entorno web.
- **Enfoque:** mixto, con predominio cualitativo-formativo y un componente cuantitativo descriptivo y exploratorio.
- **Marco metodológico:** DSR para construir y evaluar el artefacto, complementado con observación cualitativa de uso, cuestionarios estandarizados y medición de KPIs técnicos.
- **Unidad de análisis:** la interacción de usuarios con un prototipo web de visualización técnica del dron Holybro X500 V2, contrastada con un soporte 2D de referencia para tareas equivalentes.

Marco de Design Science Research

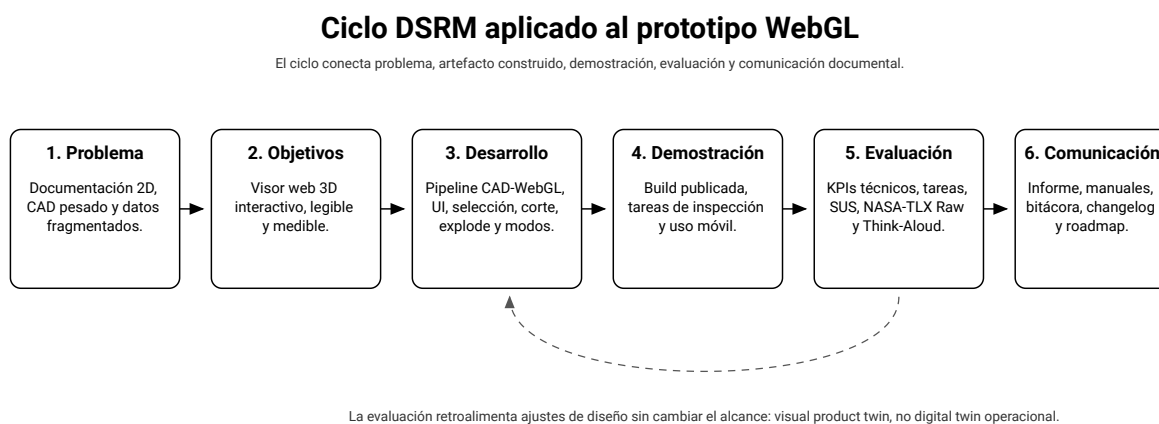
La investigación se enmarcó en la epistemología de *Design Science Research*. En términos procedimentales, el proyecto adoptó el modelo de seis fases del DSRM propuesto por Peffers et al. (2007), mientras que la evaluación de rigor y relevancia del artefacto se fundamentó en las siete directrices establecidas por Hevner et al. (2004). De este modo, el proceso de construcción y evaluación del artefacto quedó diferenciado de las pautas con las que se juzga su consistencia metodológica y su aporte disciplinar. La Figura 14 sintetiza esa aplicación del ciclo DSRM al caso del visor WebGL.

1. **Identificación del problema y motivación.** La documentación 2D tradicional dificulta la comprensión espacial y funcional de sistemas de hardware con múltiples componentes; además, la industria suele mantener separados CAD, manuales, listas de materiales, procedimientos, sensores y conocimiento experto.
2. **Definición de objetivos de la solución.** Desarrollar un visor web interactivo orientado a facilitar la exploración y lectura técnica del dron Holybro X500 V2 bajo restricciones reales de rendimiento, entendiendo el prototipo como una capa visual-semántica previa a posibles fases de *digital shadow*.
3. **Diseño y desarrollo.** Construir el prototipo en Unity con un *pipeline* CAD a web, UI responsiva y herramientas de análisis visual.
4. **Demostración.** Publicar el prototipo en entorno web y ejecutar tareas representativas de exploración, identificación y análisis.
5. **Evaluación.** Analizar de forma formativa el rendimiento técnico, la usabilidad percibida y la carga de trabajo percibida del prototipo.

6. **Comunicación.** Documentar el proceso, la arquitectura, las decisiones de diseño y los resultados obtenidos.

Figura 14

Aplicación del ciclo Design Science Research al proyecto WebGL.



Nota. La evaluación retroalimenta ajustes del artefacto, pero no cambia el alcance metodológico: se evalúa un *visual product twin*, no un gemelo digital operacional. Elaboración propia.

Delimitación metodológica como visual product twin

El artefacto evaluado en este informe se delimita como *visual product twin* o modelo visual-semántico de producto, no como *digital twin* operacional. Esta delimitación metodológica evita evaluar el sistema con criterios que no corresponden a su alcance actual. Por tanto, la validación se concentra en viabilidad web, comprensión espacial, inspección visual, usabilidad percibida, carga de trabajo subjetiva y coherencia funcional de la interfaz. No se evalúan sincronización con activo físico, mantenimiento predictivo, integración PLM/CMMS/SCADA ni modelos calibrados de simulación, porque esas capacidades pertenecen al roadmap futuro.

Esta decisión también orienta los instrumentos. Los KPIs técnicos y las pruebas de usuario permiten juzgar si la capa visual-semántica es funcional, comprensible y defendible como

base de trabajo; no permiten afirmar que el sistema sea una sombra digital o un gemelo digital.

Para evolucionar a esas categorías serían necesarios, como mínimo, un *twin manifest* por pieza, telemetría histórica o en vivo, trazabilidad de eventos, reglas de mantenimiento y validación de datos operacionales (véase la Figura 15).

Figura 15

Frontera metodológica entre lo evaluado y lo proyectado.

Frontera Metodológica: Evaluado vs. Proyectado

Característica	Evaluado	Proyectado
Alcance	Viabilidad web, navegación 3D, selección, aislamiento, fichas de pieza, modos visuales, corte, vista explosionada, rendimiento, usabilidad percibida, carga de trabajo subjetiva, coherencia de la interfaz, utilidad formativa observada	Sincronización con activo físico, telemetría real, mantenimiento predictivo, integración PLM/CMMS/SCADA/IoT, diagnóstico operacional, modelos calibrados de simulación
Elementos Clave	WebGL, visualización, interacción, rendimiento, SUS, NASA-TLX, Think-Aloud	Telemetría, sincronización, mantenimiento, simulación, digital twin
Tipo de Twin	Visual-semántico	Digital shadow, digital twin

Nota. Las capacidades de telemetría, sincronización, mantenimiento predictivo y simulación calibrada se mantienen fuera de la evaluación actual. Elaboración propia.

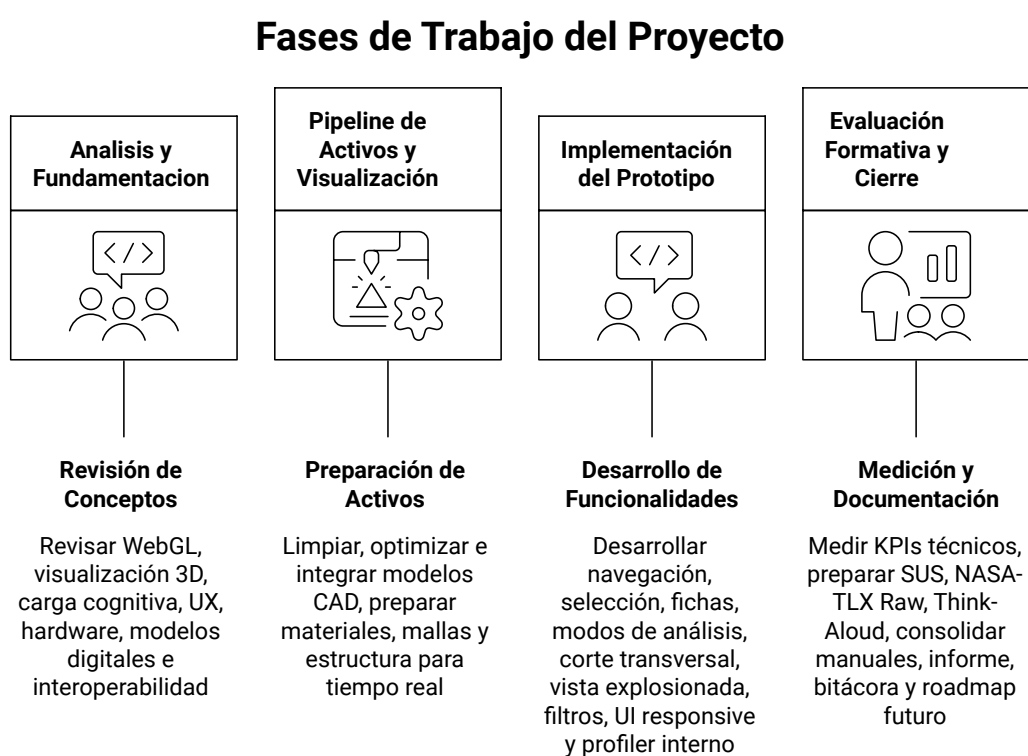
Fases de trabajo

1. **Análisis y fundamentación.** Revisión bibliográfica sobre WebGL, visualización 3D, teoría de la carga cognitiva, carga de trabajo percibida, UX, hardware complejo, *digital model*, *digital shadow*, *digital twin* e interoperabilidad; selección del caso de estudio; definición de metas de rendimiento y alcance funcional.
2. **Pipeline de activos y visualización.** Limpieza, optimización e integración del modelo CAD; preparación de materiales y mallas para visualización en tiempo real; configuración del proyecto Unity y del *pipeline* web.
3. **Implementación del prototipo.** Desarrollo de interacciones, selección de piezas, flujo de detalle, modos de análisis visual, vista explosionada, corte transversal e interfaces responsivas.
4. **Evaluación formativa y cierre.** Medición de KPIs técnicos, aplicación de SUS, NASA-TLX Raw y *Think-Aloud*, consolidación del informe, manuales y anexos, y formulación de un roadmap de evolución hacia *twin manifest*, telemetría histórica, *live digital shadow* y modo de servicio (véase la Figura 16).

Prueba breve formativa previa a la evaluación final

En este informe, las etiquetas P1 y P2 se usan como niveles internos de prioridad de QA: P1 agrupa correcciones críticas para que el flujo principal sea usable y defendible; P2 agrupa ajustes importantes de pulido, consistencia o reducción de fricción que no bloquean por sí solos la demostración del prototipo.

Antes de aplicar los instrumentos con la muestra final, se realizó una prueba breve exploratoria con usuarios para observar fricciones tempranas de interacción. Esta actividad se consi-

Figura 16*Fases de trabajo del proyecto.*

Nota. Las fases agrupan fundamentación, preparación de activos, implementación y cierre evaluativo-documental.
Elaboración propia.

dera una iteración formativa previa, alineada con el enfoque *Think-Aloud* y con observación de acciones espontáneas o inconscientes, y permitió ajustar la versión que luego fue evaluada con SUS, NASA-TLX Raw y protocolo comparativo 3D/2D.

El objetivo de esta prueba previa fue identificar gestos que los usuarios intentaban realizar de manera automática antes de recibir instrucción detallada. Entre los patrones observados se registraron intentos de hacer *swipe* sobre las tarjetas del tutorial, uso de gestos verticales para abrir o cerrar el panel de información, sensibilidad excesiva percibida en el zoom táctil, expectativa de doble toque para aislar elementos y rechazo a la apertura automática del panel de información cuando la intención principal era aislar una pieza. Estos hallazgos alimentaron los ajustes de cierre de la interfaz para hacer que la aplicación respondiera mejor a expectativas naturales de uso en la evaluación posterior.

La revisión formativa produjo una lista de correcciones priorizadas que se trabajó antes del cierre de QA de P1 y P2. En particular, se ajustaron los gestos del tutorial, la apertura y cierre del *bottom sheet*, el bloqueo táctil del panel de información para evitar interacción accidental con el dron, la sensibilidad de zoom y órbita, el doble toque sobre piezas y hotspots, la navegación por capas con el botón Atrás del navegador, la eliminación de la X externa del contenedor WebGL y la estabilización de iconos y microinteracciones en móvil. Por tanto, esta prueba breve se reporta como evidencia formativa de diseño iterativo y se distingue de los resultados con usuarios consolidados en el capítulo 5 (véase la Figura 17).

Muestra y perfil de participantes

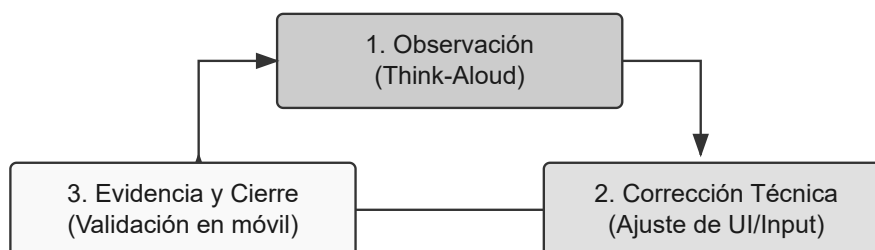
Se proyectó una muestra no probabilística por conveniencia de al menos 30 participantes como escenario deseable para fortalecer la lectura descriptiva de los instrumentos estandarizados.

El estudio integrado en el informe se realizó con 12 participantes anonimizados, dentro

Figura 17

Ciclo de QA formativo y optimización de interacción.

Ciclo de QA Formativo y Ajuste de Interacción



Nota. Proceso iterativo de detección de fricciones de interfaz y su correspondiente ajuste funcional en la build. Elaboración propia.

del escenario mínimo operativo definido para evaluación formativa. El perfil esperado corresponde a estudiantes avanzados, técnicos o profesionales de ingeniería, manufactura digital o mantenimiento. Esta decisión se apoyó en la tradición de evaluaciones formativas en HCI y UX, donde muestras reducidas permiten identificar problemas recurrentes de interacción y generar insumos de rediseño (Nielsen, 2000; Lazar et al., 2017). En consecuencia, los resultados de SUS y NASA-TLX Raw se interpretan de forma descriptiva y exploratoria, sin pretensión de inferencia estadística poblacional.

La caracterización efectiva de la muestra se reporta de forma agregada para preservar el anonimato de los participantes y evitar la asociación directa entre perfil profesional y respuestas individuales. La muestra quedó conformada por perfiles afines a ingeniería, desarrollo de software, diseño de interacción y visualización 3D, coherentes con el carácter técnico-formativo de la evaluación (véase la Tabla 2).

Tabla 2*Perfil académico o profesional agregado de la muestra*

Perfil académico o profesional	Cantidad
Ingeniería de sistemas, estudiantes o profesionales	3
Desarrollo de software	1
Ingeniería multimedia	1
Ingeniería mecatrónica	1
Ingeniería civil	1
Ingeniería electrónica	2
Ingeniería mecánica	1
Diseño gráfico con énfasis en UX/UI	1
Entusiasta 3D	1
Total	12

Nota. La tabla presenta datos agregados de caracterización y no permite asociar perfiles con respuestas individuales. Elaboración propia.

Variables de investigación

Variable independiente

Tipo de visualización técnica

- **Definición conceptual:** medio utilizado para consultar y comprender la información del sistema.
- **Definición operacional:** comparación entre el visor 3D interactivo en web y un soporte 2D de referencia.
- **Niveles:** visor 3D interactivo / documentación 2D.

Variables dependientes

Usabilidad percibida

- **Definición conceptual:** grado en que el usuario percibe el sistema como fácil de aprender, consistente y útil.

- **Definición operacional:** puntaje total del cuestionario *System Usability Scale* (Brooke, 1996), aplicado únicamente al prototipo 3D interactivo y expresado en escala de 0 a 100.
- **Criterio de interpretación:** 68 se toma como referencia del promedio histórico del instrumento y no como sinónimo de usabilidad alta. Como guía contextual, puntajes en el rango de 70 a 72 se interpretan como favorables o por encima del centro de referencia, mientras que puntajes en los ochenta corresponden a niveles claramente superiores de aceptación. Estas lecturas se usan solo como contextualización descriptiva y se respaldan en Bangor et al. (2008, 2009) y Sauro y Lewis (2016), no como umbrales absolutos de validación.

Carga de trabajo percibida

- **Definición conceptual:** percepción subjetiva de demanda y esfuerzo durante la ejecución de tareas, incluyendo demanda mental, demanda temporal, esfuerzo, frustración y percepción de rendimiento.
- **Definición operacional:** promedio de las seis dimensiones del NASA-TLX Raw (Hart & Staveland, 1988), en escala de 0 a 100.
- **Criterio de interpretación:** describir la magnitud de la carga de trabajo percibida y contrastar si el prototipo 3D tiende a producir una carga de trabajo igual o menor que el soporte 2D de referencia en tareas comparables. En este informe, NASA-TLX no se interpreta como una medición directa de las categorías intrínseca, extrínseca y germana de la Teoría de la Carga Cognitiva, sino como un indicador de carga de trabajo subjetiva percibida durante la tarea.

Desempeño en tareas de inspección

- **Definición conceptual:** capacidad del usuario para completar tareas representativas de identificación, ubicación y consulta técnica sobre el ensamblaje.
- **Definición operacional:** tiempo en segundos para T1, T2 y T3, tasa de tareas completadas, completitud autónoma, número de errores observados, solicitudes de ayuda y observaciones cualitativas registradas por condición.
- **Criterio de interpretación:** se considera más favorable la condición que permita resolver las tareas con menor tiempo medio en T1–T3, mayor porcentaje de completitud, menor intervención del moderador y verbalizaciones que indiquen mejor orientación durante la inspección.

Rendimiento técnico y uso de memoria

- **Definición conceptual:** desempeño del prototipo web durante la carga y la interacción en tiempo real, junto con el comportamiento de memoria asociado al despliegue en navegador.
- **Definición operacional:** medición de FPS, *frame time*, primera captura operativa del profiler, lotes renderizables o *draw calls* de referencia cuando estuvieran disponibles, presupuesto geométrico, tamaño de artefactos de *build* relevantes para la carga inicial (especialmente archivos *.data* y asociados), tamaño del *heap* configurado para Web, crecimiento efectivo del *heap* durante la sesión y memoria observada del runtime en navegador, acompañados del registro explícito del equipo, navegador, resolución y escenario técnico.
- **Metas operativas de referencia:** 30 FPS o más y *frame time* promedio igual o inferior a 33,33 ms en el entorno de prueba controlado definido para la validación. Los indicadores de memoria se reportan de forma trazable y contextualizada, evitando re-

ducirlos a una única cifra desanclada del dispositivo y del navegador.

Variables de control y condiciones registradas

- código anónimo del participante, fecha, perfil académico o profesional agregado, experiencia previa declarada, dispositivo de uso, sistema operativo, navegador y resolución;
- condición aplicada en cada bloque de prueba: visor 3D o soporte 2D de referencia;
- orden de exposición a las condiciones 3D y 2D;
- versión de build, estado de caché y URL o archivo utilizado durante la evaluación;
- para mediciones técnicas WebGL: dispositivo, sistema operativo, navegador, resolución, escenario, versión de build y archivo JSON/CSV exportado por el profiler interno.

El entorno de prueba controlado quedó delimitado por la misma build usada en la recolección de datos técnicos y en la evaluación con usuarios. Para la copia pública de cierre, la trazabilidad se fija en la rama `master` del repositorio y en los artefactos versionados en `docs/Build`, `Informe_final/` y `Telemetria/Mediciones_WebGL`. La build se ejecutó con la caché del navegador previamente cargada. En escritorio se usó el mismo equipo empleado en las pruebas técnicas: Intel Core i7-5820K, NVIDIA GTX 980 Ti, 48 GB RAM, resolución efectiva del viewport de 1910×903, Windows 11 y Google Chrome 141. En móvil se utilizó un Redmi Note 10S con MIUI Global 14.0.11 y Google Chrome 148. Estas condiciones se registran para asegurar trazabilidad y evitar extrapolar el comportamiento observado a equipos o configuraciones no evaluadas. Cuando los archivos exportados por Unity WebGL reportan sistemas como `Windows 10` o `Android 10`, ese valor se interpreta como metadato normalizado del navegador o del run-

time, y se prioriza el registro manual del dispositivo, sistema operativo y navegador consignado por el moderador.

Diseño comparativo y condiciones de prueba

La evaluación con usuarios se planteó como un diseño intra-sujeto de carácter formativo. Cada participante interactuó con dos condiciones funcionalmente equivalentes: **Condición A**, correspondiente al prototipo 3D interactivo en navegador, y **Condición B**, correspondiente a un soporte 2D de referencia construido a partir de documentación técnica estática del mismo sistema. La comparación no pretendió probar causalidad psicológica concluyente, sino describir diferencias observables de percepción y uso entre ambos medios.

Definición operativa de la condición 2D. La condición 2D se implementó mediante un paquete documental controlado, fijo para todos los participantes, derivado de los manuales del fabricante —*X500 Kit Assembly Manual* y *Holybro X500 Frame Kit Assembly Guide*— y complementado con una lámina de rotulado alineada con la taxonomía canónica del proyecto (Holybro, s. f.-a, s. f.-b).

Para mantener equivalencia informativa con la condición 3D, dicho paquete incluyó una vista general del ensamblaje, una secuencia o despiece del frame, una lámina de identificación de subsistemas de propulsión y electrónica, y una ficha textual de componentes por categoría. Durante la prueba, los participantes consultaron únicamente ese paquete estático mediante lectura, desplazamiento o cambio de página, sin rotación 3D, filtros interactivos ni herramientas analíticas propias de la condición 3D.

Para mitigar sesgos de aprendizaje o fatiga, el orden de exposición se contrabalanceó mediante dos secuencias posibles: AB y BA. La asignación del orden se realizó por alternancia simple: los participantes con código impar siguieron la secuencia AB, es decir, visor 3D primero y

sopORTE 2D después; los participantes con código par siguieron la secuencia BA, es decir, sopORTE 2D primero y visor 3D después. Esta regla permitió distribuir de forma controlada el posible efecto de aprendizaje o fatiga sin introducir una asignación subjetiva por parte del moderador. Las tareas mantuvieron el mismo objetivo funcional en ambas condiciones y variaron únicamente en el medio de representación. El moderador empleó instrucciones equivalentes y una misma matriz de registro para tiempo en T1–T3, éxito, errores, ayudas y observaciones.

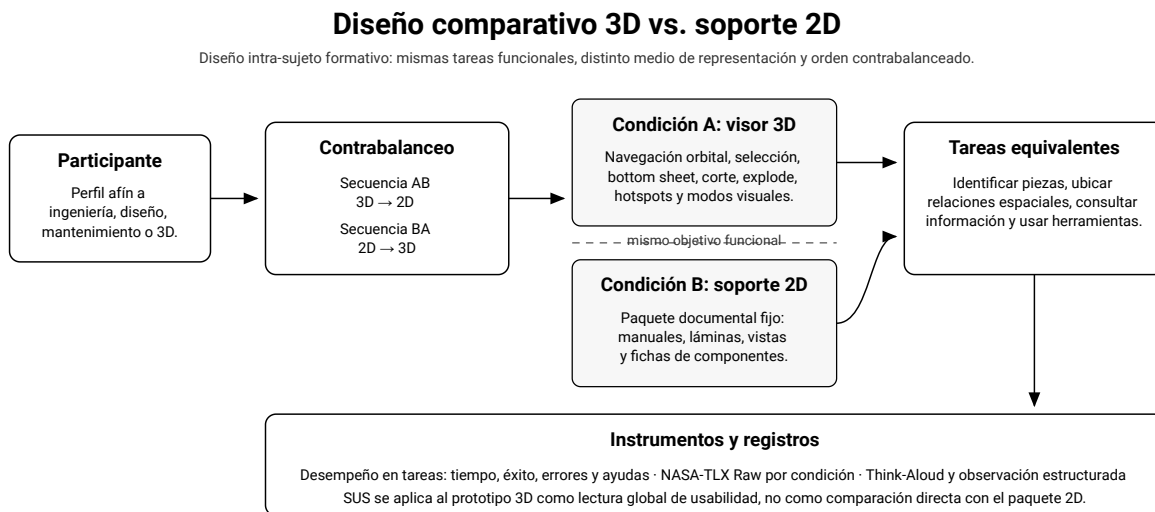
El entorno técnico se documentó de forma explícita en las sesiones de rendimiento WebGL, incluyendo equipo, sistema operativo, navegador, resolución, escenario evaluado y archivo exportado por el profiler interno. Los KPIs técnicos de la build web se interpretan junto con esas condiciones de ejecución para evitar extrapolaciones ambiguas del desempeño.

En cuanto al alcance de plataforma, el prototipo se concibió para ejecutarse en navegadores web compatibles de escritorio y móviles soportados por la plataforma web de Unity. La compatibilidad efectiva depende del navegador, su versión, la memoria disponible y el nivel de optimización de la build. Por ello, la evaluación técnica registró dispositivo, navegador, resolución, escenario y medidas fechadas, evitando extrapolar el comportamiento observado a equipos no incluidos en la prueba (Unity Technologies, s. f.).

Una mejora operativa de la condición 3D frente a la referencia 2D se define por la convergencia descriptiva de varios indicadores: mayor porcentaje de éxito en tareas, menos errores o ayudas requeridas, carga de trabajo percibida igual o inferior y verbalizaciones cualitativas que indiquen mejor orientación durante la inspección. El SUS se utiliza por separado como lectura global de usabilidad del prototipo 3D y no como métrica comparativa entre condiciones (véase la Figura 18).

Figura 18

Diseño comparativo entre condición 3D interactiva y soporte 2D de referencia.



Nota. El diseño usa tareas equivalentes y orden contrabalanceado; SUS se reporta para el prototipo 3D como lectura global de usabilidad. Elaboración propia.

Instrumentos

- **KPIs técnicos:** Unity Profiler mediante *Development Build* y *Autoconnect Profiler* cuando corresponda, herramientas del navegador, auditorías de cobertura y jerarquía, y el profiler interno de la aplicación WebGL para capturar sesiones por escenario desde la propia build.
- **Registro de desempeño en tareas:** matriz estructurada por condición para documentar tiempo en segundos para T1, T2 y T3, porcentaje de éxito, completitud autónoma, errores observables, ayudas requeridas y observaciones.
- **SUS:** cuestionario posterior a la interacción con el prototipo 3D para estimar usabili-

- dad percibida mediante 10 ítems en escala Likert de 1 a 5.
- **NASA-TLX Raw:** cuestionario posterior a tareas para estimar carga de trabajo percibida en seis dimensiones, cada una en escala de 0 a 100. Se usa como medida de carga de trabajo percibida y no como medición directa de la Teoría de la Carga Cognitiva. En este informe se adopta la variante *Raw TLX* por su pertinencia para lectura formativa rápida sin ponderación pareada adicional (Hart & Staveland, 1988; Hart, 2006).
 - **Think-Aloud:** protocolo de verbalización concurrente con instrucción estandarizada, intervención neutral del moderador y apoyo de matriz de observación, audio y captura de pantalla cuando el entorno lo permita (véase la Figura 19).

Figura 19

Instrumentos de evaluación del prototipo.

Instrumentos de Evaluación del Prototipo

Característica	KPIs Técnicos	Registro de Desempeño en Tareas	SUS	NASA-TLX Raw	Think-Aloud
Propósito	Mide la viabilidad de la build	Mide el uso operativo	Mide la usabilidad percibida	Mide la carga de trabajo subjetiva	Recoge verbalizaciones
Métricas	FPS, frame time, memoria, carga, escenario, dispositivo, navegador, resolución	Tiempo, éxito, errores, ayudas requeridas	Diez ítems, puntaje de 0 a 100	Demanda mental, física, temporal, rendimiento, esfuerzo, frustración	Verbalizaciones, dudas, estrategias, errores, percepciones
Capa	Técnica	Conductual	Perceptiva	Perceptiva	Cualitativa

Nota. Los instrumentos se organizan por capas técnica, conductual, perceptiva y cualitativa para evitar depender de un único indicador. Elaboración propia.

Protocolo de tareas

Las tareas de validación se alinearon con el flujo visible final de la aplicación y se organizaron en cuatro bloques equivalentes para ambas condiciones:

1. **Ubicación de componente:** localizar un componente específico del dron en el visor 3D o en el soporte 2D.
2. **Interpretación funcional:** explicar la función general del componente y su relación con el sistema.
3. **Relación espacial:** identificar vínculos espaciales entre el componente, piezas cercanas y el ensamblaje.
4. **Análisis visual guiado:** usar una herramienta visible del visor —por ejemplo *Explode*, *Cut*, *X-Ray*, *Thermal* o presets de *Studio*— o interpretar la información equivalente en el soporte 2D.

Procedimiento de análisis

Análisis cuantitativo

El componente cuantitativo del estudio tuvo carácter descriptivo y exploratorio. La lectura cuantitativa se organizó por tipo de dato: desempeño en tareas por condición, usabilidad global del prototipo 3D, carga de trabajo percibida por condición y KPIs técnicos de la build web bajo entorno controlado.

El puntaje SUS se calcula según el procedimiento estándar de Brooke:

$$SUS = \left(\sum_{j \in \text{impares}} (R_j - 1) + \sum_{j \in \text{pares}} (5 - R_j) \right) \times 2,5$$

donde R_j corresponde a la respuesta del participante en el ítem j . Un puntaje alto indica mejor

usabilidad percibida. Para su lectura, 68 se considera una referencia histórica aproximada del instrumento, mientras que puntajes en el rango 70–72 se interpretan como favorables o por encima del centro de referencia y los valores en los ochenta como claramente superiores, siempre de forma descriptiva y no como umbral absoluto (Bangor et al., 2008, 2009; Sauro & Lewis, 2016). En este estudio, el SUS se reporta únicamente para la condición 3D.

Para NASA-TLX Raw se promedian las seis dimensiones del instrumento y se obtiene un resultado independiente para cada condición de prueba:

$$NASA-TLX_{Raw} = \frac{1}{6} \sum_{i=1}^6 D_i$$

donde D_i corresponde al puntaje observado en demanda mental, demanda física, demanda temporal, rendimiento, esfuerzo y frustración. En esta adaptación se utiliza la variante *Raw TLX*, por lo que no se aplica ponderación por comparaciones pareadas. La dimensión de rendimiento se diligencia con orientación invertida (0 = desempeño perfecto, 100 = desempeño deficiente) para conservar coherencia direccional en el promedio global. Además del promedio agregado, se reportan las subescalas para evitar interpretaciones excesivamente reductivas (Hart & Staveland, 1988; Hart, 2006).

Los KPIs técnicos se contrastaron con las metas operativas del proyecto a partir de datos obtenidos sobre la build publicada. La aplicación integra herramientas internas de captura por escenario en JSON/CSV; por ello el informe reporta FPS, *frame time*, memoria, artefactos de build, presupuesto geométrico y compatibilidad mediante tablas cuantitativas trazables.

Análisis cualitativo

El protocolo *Think-Aloud* se aplicó en modalidad concurrente durante la ejecución de tareas. A cada participante se le solicitó verbalizar qué intentaba hacer, cómo interpretaba la interfaz, qué dudas le surgían, qué errores percibía y con qué criterio decidía su siguiente acción. El evaluador registró verbalizaciones relevantes, pausas, puntos de vacilación, errores de navegación, solicitudes de ayuda y estrategias de recuperación mediante una matriz estructurada por tarea y por condición.

Posteriormente, las observaciones se codificaron por categorías analíticas: comprensión espacial del ensamblaje, navegación y control de cámara, interpretación de la interfaz, correspondencia terminológica, errores o bloqueos y sugerencias de mejora. La codificación se realizó mediante lectura de los registros del moderador, una segunda pasada de revisión y consolidación de memos breves por participante. Esta lectura cualitativa permite triangular los resultados de SUS y NASA-TLX y explicar por qué determinadas tareas resultaron más claras, ambiguas o demandantes para el usuario (Ericsson & Simon, 1993).

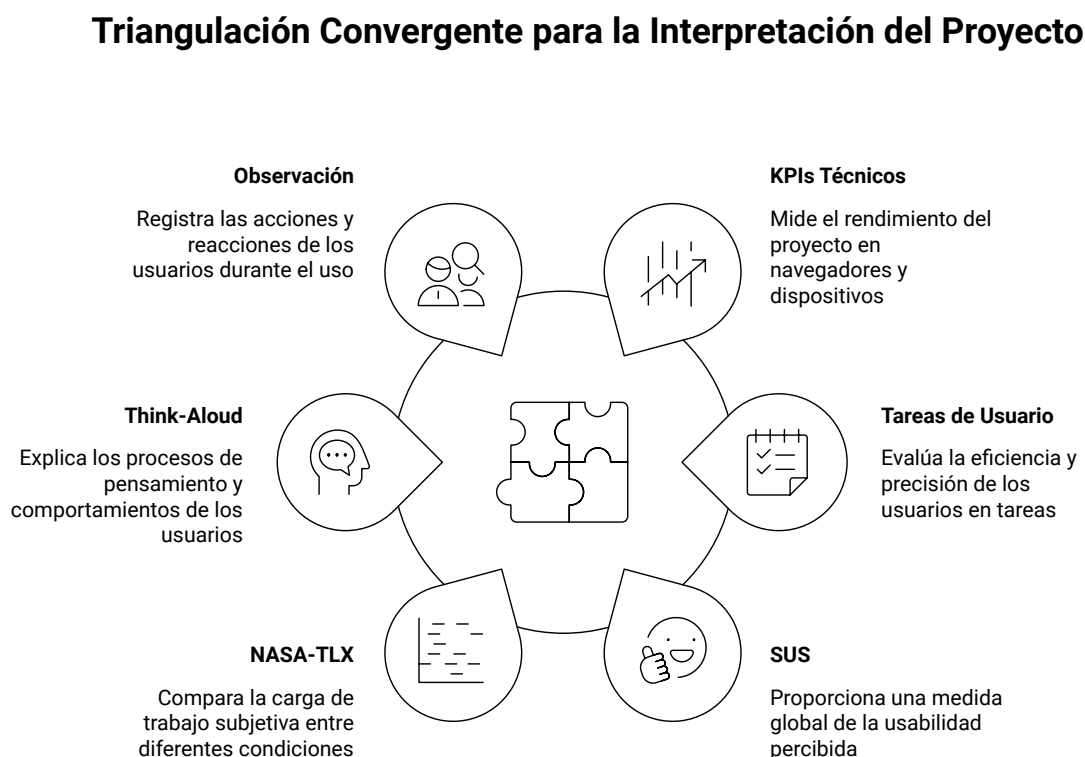
Triangulación de hallazgos

La integración final de resultados sigue una lógica de triangulación convergente. En primer lugar, los indicadores de desempeño en tareas y los KPIs técnicos describen la viabilidad operativa de la build web en el entorno de prueba documentado. En segundo lugar, SUS aporta una lectura global de usabilidad del prototipo 3D, mientras que NASA-TLX compara la carga de trabajo percibida entre la condición interactiva y la referencia 2D. En tercer lugar, el *Think-Aloud* y la observación estructurada explican por qué determinadas tareas producen fricción, claridad o ambigüedad. Un hallazgo se considera más sólido cuando muestra consistencia entre estas ca-

pas de evidencia, teniendo en cuenta que la comparación entre 3D y 2D se apoya en desempeño y carga de trabajo percibida, mientras que la usabilidad global se informa para el prototipo interactivo (véase la Figura 20).

Figura 20

Triangulación de evidencia para interpretar el prototipo.



Nota. La interpretación final combina KPIs técnicos, desempeño en tareas, SUS, NASA-TLX Raw, *Think-Aloud* y observación estructurada. Elaboración propia.

Consideraciones éticas

La participación en pruebas fue voluntaria. Los participantes fueron identificados mediante códigos anónimos y los resultados se reportan de forma agregada o anonimizada. No se recolectaron datos sensibles ni se realizaron intervenciones físicas. Cualquier registro de audio,

pantalla o fotografía quedó condicionado a autorización explícita, y los documentos de validación se conservaron únicamente como soporte académico del proyecto. El estudio no contempla exposición a riesgos adicionales para los participantes.

Desarrollo e Implementación

Este capítulo documenta el estado técnico del desarrollo: qué se implementó, qué se evaluó como parte del proceso, qué quedó validado en la build funcional y qué frentes se delimitan como trabajo futuro. Para mantener rigor documental, cada decisión de producción se clasifica en tres estados: **Ejecutado**, **Evaluado** y **Descartado o reemplazado**. La escena oficial de referencia sigue siendo `MainScene_Final`.

Estado del desarrollo

El desarrollo del proyecto no se redujo a la construcción de una arquitectura de software. También involucró decisiones acumulativas de modelado, importación CAD, automatización, UI/UX, shaders, visualización analítica, saneamiento de escenas, organización de datos y validación técnica. En consecuencia, este capítulo se organiza para explicar el producto como un *pipeline* completo de *Technical Art* y desarrollo interactivo.

La lógica de organización adoptada es la siguiente:

- **Ejecutado:** procesos, herramientas y flujos que dejaron huella verificable en la app, la escena, el código o los artefactos activos.
- **Evaluado:** rutas técnicas estudiadas, probadas parcialmente o utilizadas como soporte de decisión, aunque no constituyan el estado final consolidado.
- **Descartado o reemplazado:** enfoques tempranos que ayudaron a tomar decisiones, pero ya no representan la versión defendible del producto.

Esta distinción es especialmente importante porque el cierre del informe combina dos tipos de evidencia: por un lado, la build WebGL publicada y validada funcionalmente en los frentes de prioridad P1/P2; por otro, las tablas cuantitativas de rendimiento, SUS, NASA-TLX Raw y

desempeño en tareas, que se reportan con datos fechados dentro del capítulo de resultados. En este informe, P1 agrupa correcciones críticas del flujo principal y P2 agrupa ajustes importantes de pulido o reducción de fricción. El FBX runtime final, texturas/materiales y auditorías de reimportación se conservan como trazabilidad técnica del pipeline y no deben confundirse con resultados empíricos de usuario.

Normalización del sistema

Para evitar contradicciones entre código, escena y documentación, este informe adopta las siguientes convenciones:

- **28 piezas canónicas** de investigación reflejadas en los assets `DronePartData` generados y en el catálogo público de fichas bilingües;
- **30 anchors de escena**: las 28 piezas más `x500v2_fastener_group` y `x500v2_misc_group`;
- **257 renderers/colliders** auditados en la escena final;
- **257 assets generados** en `Assets/Core/Data/X500V2Generated`;
- inventario de scripts organizado en tres niveles de lectura: 95 archivos runtime propios, 103 archivos bajo `Assets` si se excluyen tests y 129 archivos `.cs` en total al incluir editor y plugins.

En este contexto, un *anchor* es un nodo operativo de escena usado por Unity para agrupar selección, aislamiento, filtros, vista explosionada, metadatos y jerarquía de runtime. Los 30 *anchors* no reemplazan la taxonomía semántica de 28 piezas: la extienden con dos grupos técnicos. `x500v2_fastener_group` agrupa sujetadores repetitivos, mientras que `x500v2_misc_group` reúne elementos misceláneos o auxiliares que no debían forzarse dentro

de una pieza canónica principal, como conectores, soportes menores o cubiertas.

La cifra histórica de “16 piezas” se conserva solo como antecedente temprano del proyecto. Corresponde a una etapa de prototipo en la que se simplificó el ensamblaje para validar navegación, selección, arquitectura UI y flujo de datos con un conjunto reducido. Al consolidar el Holybro X500 V2 como caso de hardware complejo, esa lectura dejó de ser suficiente: el sistema pasó a una taxonomía estabilizada de 28 piezas semánticas y a su traducción técnica en 30 *anchors* y 257 elementos renderizables (véanse la Tabla 3 y la Figura 21).

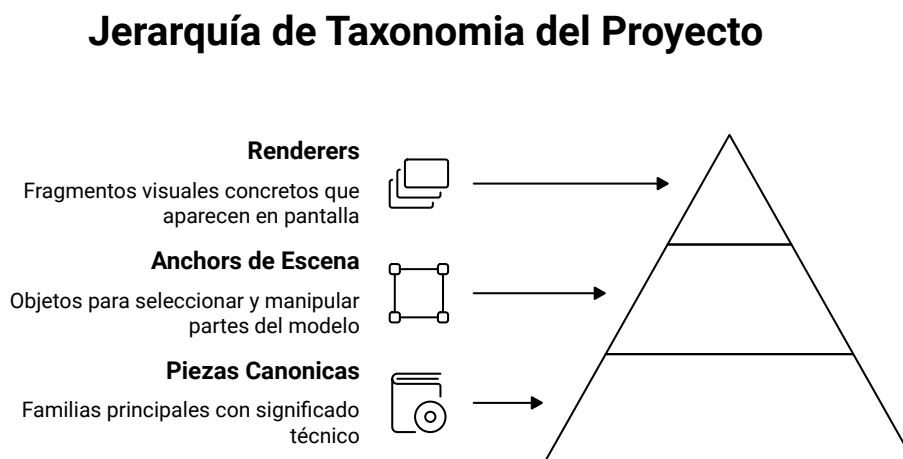
Tabla 3*Reconciliación entre conteos históricos y convenciones finales del proyecto*

Conteo	Nivel de lectura	Estado	Interpretación correcta
16 piezas	Prototipo temprano y simplificación del ensamblaje	Histórico	Útil para entender la etapa inicial del visor, pero no describe la escena ni la taxonomía final del proyecto.
28 piezas	Taxonomía semántica de investigación	Canónico	Corresponde al catálogo académico principal utilizado para narrativa, hotspots, fichas y documentación.
30 <i>anchors</i>	Organización de escena en Unity	Canónico técnico	Añade x500v2_fastener_group y x500v2_misc_group como grupos sintéticos de operación de runtime.
257 <i>renderers/colliders</i>	Fragmentación geométrica y cobertura de escena	Canónico técnico	Describe la granularidad renderizable/auditable del ensamblaje final, no una nueva taxonomía semántica.
257 <i>assets</i> generados	Capa de datos derivada del runtime	Canónico técnico	Representa la producción de DronePartData y <i>assets</i> auxiliares alineados con la escena final.

Nota. Los conteos pertenecen a capas y momentos distintos del proyecto; por ello no deben interpretarse como equivalentes. Elaboración propia a partir del catálogo canónico, la escena final y los activos generados.

Figura 21

Taxonomía del sistema: relación entre piezas semánticas, anchors y renderers.



Nota. Mapeo jerárquico que organiza las 28 piezas canónicas de la investigación en 30 anchors de escena en Unity y 257 renderers y colliders físicos en runtime. Elaboración propia.

Evolución del proyecto y consolidación del caso de estudio

El desarrollo del prototipo atravesó varias reformulaciones antes de llegar al estado actual. En una fase inicial, el producto se entendía como un visor 3D con pocas piezas funcionales y fuerte énfasis en la arquitectura de interacción. Más adelante, el proyecto se desplazó hacia un caso de estudio técnicamente más exigente: el dron Holybro X500 V2 como ejemplo de *hardware complejo*, entendido aquí como un sistema electromecánico multicomponente con integración estructural, energética, electrónica y de control, alta densidad de relaciones espaciales entre piezas y necesidad de inspección visual del ensamblaje.

Ese cambio de foco alteró tanto la producción de activos como la app (véase la Tabla 4). La transición implicó abandonar una lectura reducida del sistema, ampliar la taxonomía, rees-

estructurar la escena, redefinir el flujo de selección y replantear la interfaz desde una lógica *mobile-first*. También exigió separar lo ya visible en la build de lo que pertenece a legado técnico, funciones internas o cierre documental.

Lectura del prototipo como visual product twin

Desde el cierre conceptual del proyecto, el producto se entiende como una capa de *visual product twin*: una representación web 3D optimizada, navegable y semánticamente organizada del hardware, orientada a inspección, explicación técnica y preparación de datos. Esta lectura no cambia el alcance implementado, pero sí clarifica su valor industrial. La app final no sincroniza telemetría real ni ejecuta diagnóstico de mantenimiento; sin embargo, estabiliza identificadores visuales, categorías, grupos de escena, fichas, hotspots y herramientas de análisis que podrían mapearse en el futuro a sensores, procedimientos, eventos y repuestos.

En términos de madurez, la build publicada se ubica en el nivel de *digital model* enriquecido: representa el producto con fidelidad visual y semántica, pero no recibe aún datos del activo físico. El siguiente paso lógico no sería declarar un gemelo digital completo, sino formalizar un *twin manifest* independiente de Unity y conectar datos históricos o simulados que permitan reproducir estados por pieza. Esta progresión reduce riesgo técnico, evita sobrepromesas y mantiene coherencia con la literatura que distingue *digital model*, *digital shadow* y *digital twin* (Kritzinger et al., 2018; Digital Twin Consortium, 2020). Los renderes finales del dron optimizado se reservan para el capítulo de resultados, donde se relacionan directamente con la reducción poligonal y el cierre visual del activo.

Tabla 4*Hitos de evolución técnica del desarrollo del proyecto*

Etapas	Foco principal	Estado	Lectura para el informe
Prototipo tem- prano	Arquitectura base del visor y conteo reducido de piezas	Histórica	Explica el antecedente de “16 piezas”, pero no representa el cierre documental.
Consolidación del caso Holybro	Selección del X500 V2 y ampliación de taxonomía semántica	Ejecutado	Fija el marco real del proyecto y el cambio hacia <i>hardware</i> <i>complejo</i> .
Saneamiento de activos y escena	Reorganización de pie- zas, grupos sintéticos y reparación de runtime	Ejecutado	Sustenta la convención 28/30/257 y el flujo final de la app.
Rediseño UI <i>mobile-first</i>	Bottom sheet, navegación inferior, modos y respon- sive base	Ejecutado	Aporta la capa de diseño con respaldo teórico más fuerte del producto.
Cierre técnico	Build WebGL, QA P1/P2, profiler interno y validación empírica	Build funcional y medición em- pírica	El flujo visible quedó validado para entrega; las mediciones formales se reportan con da- tos fechados de rendimiento y usuarios.

Nota. Esta tabla resume la evolución del proyecto y distingue cifras o decisiones tempranas frente al estado técnico vigente. Elaboración propia a partir del código, la documentación activa y la propuesta corregida.

Pipeline 3D, importación CAD y saneamiento geométrico

El *pipeline* de activos combinó ingeniería inversa visual, importación CAD, remodelado hard-surface, limpieza topológica, modularización de repetitivos y adaptación progresiva a un objetivo WebGL. El objetivo no fue conservar intacta la geometría de origen, sino traducirla a un activo navegable, seleccionable, explicable y desplegable en navegador bajo restricciones reales de memoria, *draw calls* y presupuesto geométrico.

Métodos de importación CAD y criterio de uso

Durante el proyecto se estudiaron y utilizaron múltiples rutas de entrada para convertir información CAD en geometría útil para Blender y Unity. La importación directa fue útil para inspección rápida, apertura inicial o comparación de piezas, pero no constituyó la ruta de mayor calidad para producción. En la práctica, la selección entre Blender directo, STEPper, MoI3D, retopología, remodelado o proxies dependió del control de teselación, la limpieza de la malla, la criticidad visual y la viabilidad WebGL de cada familia geométrica (véanse la Tabla 5 y la Figura 22).

Las Figuras 23 y 24 complementan el flujo anterior con evidencia visual de dos rutas de entrada al pipeline. En ambos casos se muestra la vista sombreada junto a su lectura *wireframe*, porque el problema técnico no se limita a la apariencia externa del dron, sino a la estructura de la malla que luego debe limpiarse, triangularse u optimizarse.

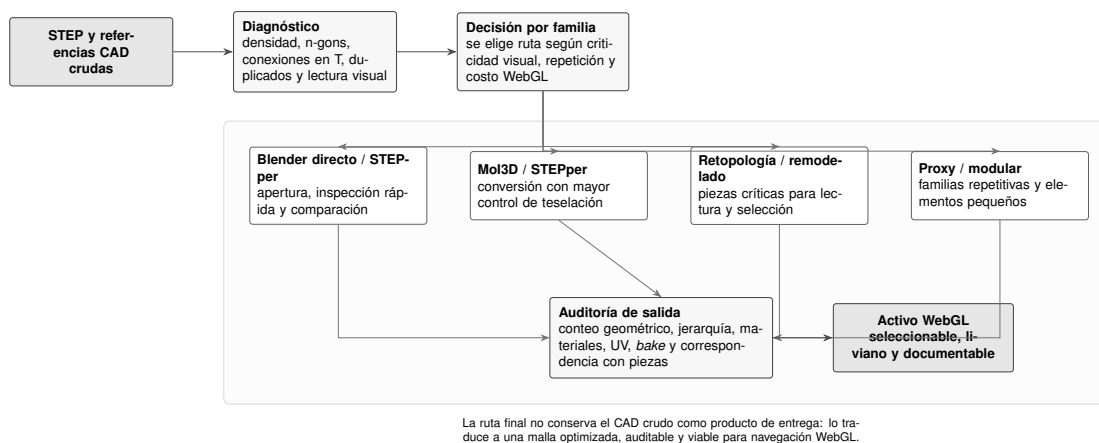
MAD-T como marco técnico adaptado para activos hard-surface WebGL

Como referencia de producción hard-surface se adoptó de manera adaptada el marco MAD-T divulgado por Blender Bros, asociado a las operaciones de *Modeling*, *Automation*, *Decimation* y *Triangulation* (Blender Bros, 2022). En su contexto original, MAD-T funciona como

Tabla 5*Métodos de importación CAD considerados en el proyecto*

Ruta técnica	Uso principal	Estado	Interpretación
STEP → Blender directo	Inspección rápida de piezas y lectura geométrica inicial	Evaluado y usado de forma exploratoria	Útil para abrir piezas o familias sencillas, pero no siempre ofrece control suficiente sobre la teselación final.
STEP → Blender mediante STEPper	Importación CAD directa dentro de Blender con apoyo de add-on especializado	Ejecutado / evaluado	Aportó una ruta práctica para abrir archivos STEP sin salir del entorno de modelado, especialmente durante revisión, limpieza o comparación de piezas.
STEP → MoI3D → exportación poligonal → Blender	Conversión de superficies NURBS/CAD con control más fino de salida	Ejecutado	Se utilizó para casos donde la importación directa producía mallas difíciles de reutilizar o inconsistentes en topología.
Re-modelado sobre referencia CAD en Blender	Reconstrucción de piezas con topología defendible para tiempo real	Ejecutado	Fue la ruta dominante para piezas críticas, especialmente cuando la malla de origen no era apta para WebGL.
Sustitución modular de repetitivos	Tornillería, arandelas y elementos similares de alta repetición	Ejecutado	Redujo redundancia geométrica y abrió la puerta a instanciación, familias y proxies ligeros.

Nota. La coexistencia de varias rutas refleja la necesidad de adaptar el pipeline a distintos tipos de geometría CAD. Elaboración propia.

Figura 22*Flujo de decisión para importación y optimización CAD a WebGL.*

Nota. Rutas técnicas de importación evaluadas y ejecutadas para trasladar la geometría CAD a Blender y Unity, indicando criterios de optimización poligonal y control de teselación. Elaboración propia.

Figura 23

Importación directa STEP a Blender con vista sombreada y wireframe.



Nota. A la izquierda se observa el modelo importado de forma directa desde STEP a Blender; a la derecha se presenta su equivalente en *wireframe*. Elaboración propia.

Figura 24

Importación desde MoI3D sin triangulación final con vista sombreada y wireframe.



Nota. A la izquierda se muestra el modelo convertido desde MoI3D sin triangular; a la derecha se evidencia la presencia de n-gons y estructura de malla que requiere saneamiento antes del uso en WebGL. Elaboración propia.

un flujo práctico para preparar mallas hard-surface de tiempo real: primero se resuelve la forma del objeto, luego se automatizan operaciones técnicas repetitivas sobre aristas, costuras y superficies, después se reduce la complejidad geométrica y finalmente se triangula de forma explícita para cerrar una malla estable.

En este proyecto, MAD-T no se usó como metodología académica ni como receta cerrada. Se utilizó como criterio técnico de producción dentro del pipeline CAD–Blender–Unity, porque el punto de partida no era un asset de videojuego modelado desde cero, sino geometría CAD heterogénea del Holybro X500 V2. Esa diferencia obligó a adaptar cada fase según el estado real de las piezas.

- **Modeling.** La fase de modelado no consistió únicamente en remodelar sobre referencia CAD. Dependiendo de la pieza, se siguieron tres rutas: piezas inexistentes que debieron modelarse desde cero; piezas existentes cuya malla CAD convertida no era trabajable y requirió retopología (véase la Figura 27); y piezas que conservaron su geometría base, pero necesitaron correcciones puntuales por problemas típicos de conversión STEP a malla, como vértices no soldados, vértices repetidos, conexiones poligonales en T, caras internas, normales inconsistentes, geometría suelta o triangulaciones inestables.
- **Automation.** Esta fase conservó el sentido técnico del MAD-T original: reducir trabajo repetitivo mediante herramientas y operaciones automatizables sobre la malla. En el proyecto incluyó selección de aristas duras, tratamiento de bevels generales, control de costuras, limpieza de marcas problemáticas, normalización de bordes, uso de modificadores y preparación consistente de superficies hard-surface antes de la reducción y

exportación.

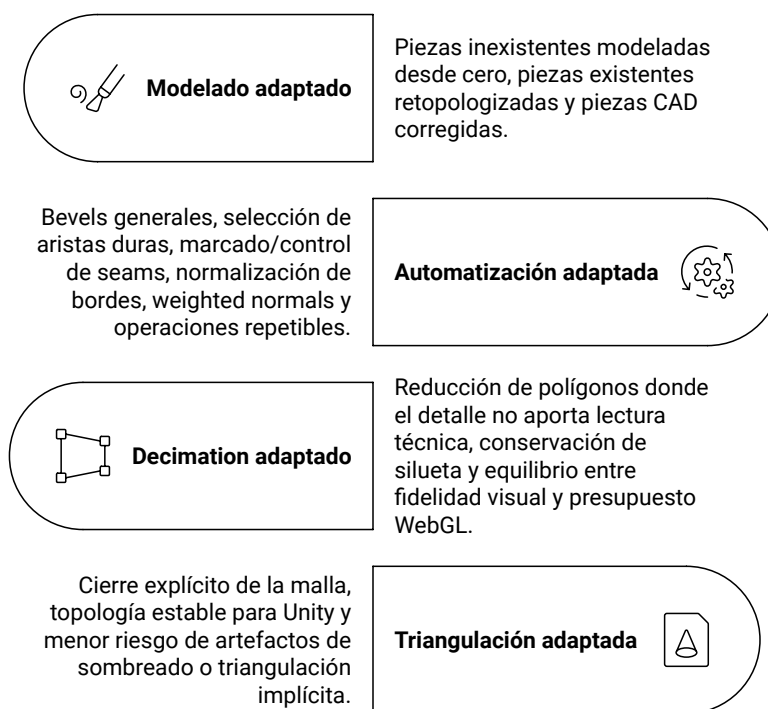
- **Decimation.** La reducción geométrica se aplicó cuando el detalle no aportaba legibilidad técnica o excedía el presupuesto esperado para WebGL. La decisión no fue eliminar detalle de forma indiscriminada, sino conservar silueta, lectura funcional y relaciones de ensamblaje mientras se disminuía el costo de vértices y triángulos.
- **Triangulation.** La triangulación se entendió como cierre de producción: una forma de evitar que Blender, el importador de Unity o el runtime WebGL decidieran triangulaciones implícitas distintas. Esto permitió estabilizar sombreado, normales, UVs y comportamiento de exportación.

Así, MAD-T operó como una capa de saneamiento y preparación de activos hard-surface (véase la Tabla 6), mientras que la metodología académica del proyecto se mantuvo en el diseño, implementación y evaluación del artefacto interactivo (véase la Figura 25).

Figura 25

Adaptación del flujo MAD-T al pipeline CAD–WebGL del proyecto.

Pasos de adaptación de MAD-T



Nota. La adaptación se usó como criterio técnico de producción hard-surface; no reemplaza la metodología académica DSRM del proyecto. Elaboración propia.

Tabla 6*Trazabilidad representativa entre familia CAD, acción aplicada y salida runtime*

Familia o pieza	Entrada o referencia principal	Acción aplicada	Salida runtime o canónica
Placas del frame	Piezas CAD del kit X500 (BOTTOM-PLATE, TOP-PLATE, PLATFORM-PLAT)	Limpieza, renombrado y adaptación para escena.	anchors canónicos x500v2_bottom_plate, x500v2_top_plate y x500v2_platform_board.
Brazos y soportes de motor	Tubo de carbono, montajes y clamps del frame.	Agrupación por familia e instancia- ción por cuadrante para conservar simetría y reducir trabajo manual.	anchors x500v2_arm_FL/FR/BL/BR.
Pixhawk 6C	Carcasa, PCB e IMU separados en CAD o referencia técnica.	Encapsulado de subpiezas en una lectura semántica única para la ficha técnica.	Anchor x500v2_pixhawk6c.
GPS M10	Antena, mástil, base, bandeja y tuerca.	Unión semántica de subpartes y normalización de nombres para selección y documentación.	Anchor x500v2_gps_m10.
Motores	Modelo base de motor y réplica por cuadrante.	Instanciación del modelo base y asignación a las posiciones funcio- nales del dron.	anchors x500v2_motor_FL/FR/BL/BR.
ESC, hélices, batería y radio	Manuales, referencias CAD par- ciales y proxies de trabajo.	Modelado o proxy según disponi- bilidad de referencia, importancia visual y tiempo disponible.	Nodos canónicos con cobertura sufi- ciente para lectura del ensamblaje.
Fasteners repetitivos	Familias CAD GB70 y sujetado- res repetidos del frame.	Clasificación semántica, proxies ligeros y reemplazo modular deli- mitado a tornillos.	Grupo x500v2_fastener_group con metadatos de familia en runtime.

Nota. La tabla resume casos representativos y no pretende listar cada subpieza del pipeline. Su función es hacer auditable la relación entre fuente geométrica, operación aplicada y destino runtime. Elaboración propia a partir de inventarios internos del pipeline, escena Blender, catálogo runtime y revisión de correspondencia CAD–Unity.

Automatización propia en Blender: masters, instancias y proxies

Además del uso adaptado de MAD-T, el proyecto requirió tooling propio dentro de Blender para resolver un problema específico de los ensambles CAD: muchas piezas repetidas llegan como objetos independientes, con transformaciones horneadas en los vértices, orígenes globales inconsistentes o mallas duplicadas que impiden aprovechar instancias reales. Esto afecta rendimiento, mantenimiento, retopología, bake de texturas y animaciones como la vista explosionada.

Para ordenar este flujo se distinguieron tres conceptos operativos:

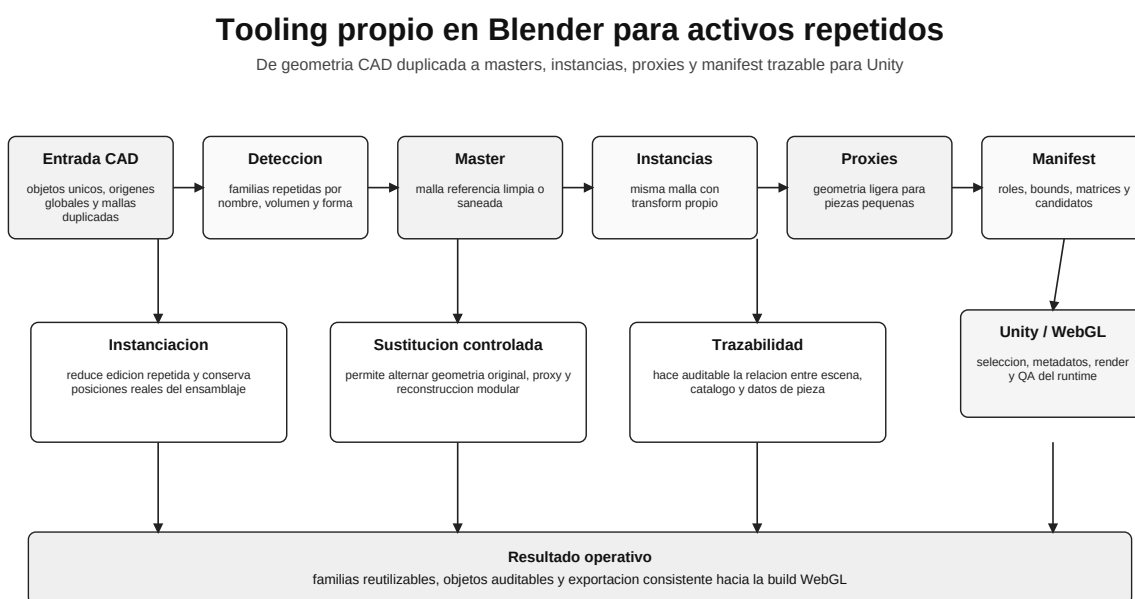
- **Master.** Malla limpia o saneada que funciona como referencia geométrica compartida de una familia repetida. Un master puede servir solo como fuente técnica para instancias o coincidir con una ocurrencia visible del ensamblaje cuando esa pieza también existe físicamente en el dron.
- **Instancia.** Objeto que comparte datos de malla con un master, pero conserva transformación, posición, rotación y escala propias dentro del ensamblaje. Esto permite corregir una sola malla y propagar el cambio a todas sus ocurrencias.
- **Proxy.** Representación geométrica ligera usada cuando una pieza no requiere detalle completo en reposo o cuando una familia repetitiva sería demasiado costosa si se conservara con su malla CAD original. El proxy conserva lectura morfológica y posición espacial, pero reduce vértices, peso y costo de inspección.

Esta separación permitió que el pipeline no dependiera de colapsar todo el modelo ni de destruir la estructura del ensamblaje. En cambio, se conservaron relaciones entre piezas físicas, instancias repetidas, geometría liviana y metadatos exportables hacia Unity. Para hacer auditable la exportación, el pipeline generó un manifest de Blender con roles, matrices de transformación,

límites espaciales, relación master/instancia y candidatos de asociación entre fasteners y pieza madre. Este manifest no reemplazó la revisión manual cuando había ambigüedad, pero redujo la posibilidad de importar una escena sin trazabilidad hacia Unity (véase la Figura 26). Los criterios usados para comparar el saneamiento geométrico se resumen en la Tabla 7; las cifras medidas corresponden al análisis de resultados.

Figura 26

Tooling propio en Blender para masters, instancias, proxies y manifest.



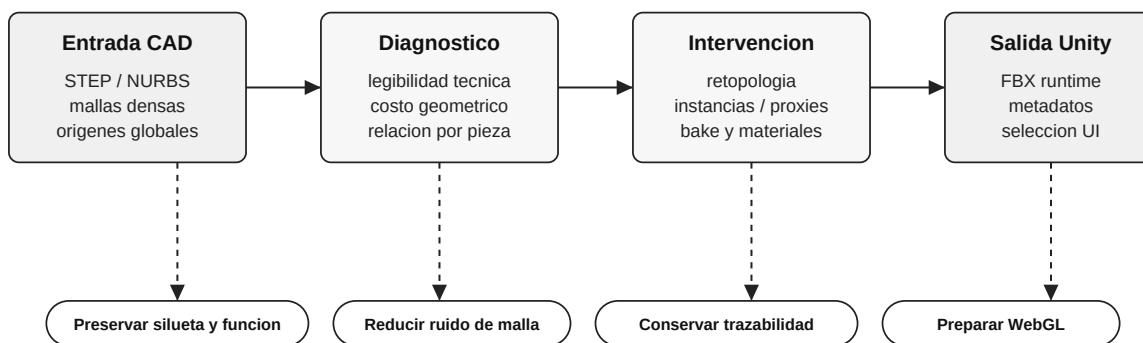
Nota. La figura muestra herramientas de preparación de activos; no corresponde a funciones visibles de la app final para usuarios. Elaboración propia.

Sistema modular de tornillería y ahorro geométrico

Uno de los problemas más costosos del pipeline fue la tornillería. Los modelos CAD tienden a repetir pernos, tuercas, arandelas y sujetadores como mallas únicas, aunque geométrica y funcionalmente pertenezcan a familias repetidas. Si esa estructura se conserva sin filtrar, se dispara el costo en vértices, *draw calls*, inspección y mantenimiento.

Figura 27

Saneamiento de geometría CAD para runtime WebGL.



Nota. Diagrama conceptual del flujo de saneamiento geométrico aplicado para transformar geometría CAD en activos aptos para runtime WebGL. Elaboración propia.

Tabla 7

Criterios para comparar el saneamiento geométrico

Familia o caso	Métrica de comparación	Lectura técnica
Brazo o soporte crítico	Triángulos, vértices y estabilidad de normales.	Permite contrastar la geometría de origen frente a una versión apta para runtime sin perder silueta funcional.
Motor o carcasa compleja	Instancias, limpieza de malla y continuidad de sombreado.	Muestra cuándo conviene conservar, limpiar o reconstruir una pieza según su visibilidad e impacto en rendimiento.
Fasteners repetitivos	Cantidad de instancias, familias y sustitución por proxy.	Evidencia el ahorro geométrico y semántico obtenido al agrupar sujetadores repetidos.
Ensamble o subsistema representativo	Costo geométrico agregado y legibilidad del ensamblaje.	Relaciona el saneamiento con presupuesto WebGL, selección por pieza y comprensión visual del sistema.

Nota. La tabla define qué se compara; los valores cuantitativos se reportan en el capítulo de resultados con evidencia de la build evaluada. Elaboración propia.

La respuesta del proyecto fue construir una capa modular para fasteners. En lugar de documentar cada tornillo como pieza semántica independiente, se organizaron **familias** reutilizables, **instancias** ubicadas en la escena y **entradas de reconciliación** para resolver diferencias de nombre entre Blender, catálogos y Unity. Una familia es el tipo reutilizable de sujetador, por ejemplo un tornillo de una métrica y forma de cabeza determinadas. Una instancia es una ocurrencia concreta de esa familia en el ensamblaje, con posición, rotación y relación espacial propia. Una entrada de reconciliación es el puente que permite asociar nombres cuando el objeto exportado desde Blender, el catálogo técnico y el identificador leído por Unity no coinciden literalmente. En la escena documentada quedaron versionadas **20 familias**, **168 instancias** y **9** entradas de reconciliación. Este trabajo alivió el presupuesto geométrico y permitió que el *runtime* mostrara metadatos de familia, métrica, longitud, material y anclaje sin duplicar la lógica por instancia (como se detalla en la Tabla 8 y se ilustra en las Figuras 28 y 29).

En este contexto, la optimización de fasteners tuvo dos niveles. El primero fue semántico: agrupar familias, registrar instancias y reconciliar nombres entre Blender, catálogos y Unity. El segundo fue geométrico: decidir cuándo conservar geometría original, cuándo usar proxies ligeros y cuándo aplicar reemplazo modular. En la versión de cierre, el sistema modular se implementó únicamente en tornillos. Otros elementos pequeños, como tuercas, standoffs, grommets o arandelas, quedaron resueltos caso por caso: algunos conservaron geometría original y otros usaron proxies ligeros cuando su detalle no aportaba información relevante para la inspección visual.

Tabla 8*Lectura técnica del sistema modular de fasteners*

Componente	Magnitud	Función en el proyecto
Familias de fasteners	20	Agrupar sujetadores equivalentes para evitar tratarlos como objetos aislados y permitir metadatos compartidos por tipo.
Instancias registradas	168	Conservan la ubicación de cada ocurrencia concreta dentro del ensamblaje y permiten asociarla con una familia.
Entradas de reconciliación	9	Resuelven diferencias entre nombres de escena, catálogo y metadatos para que Unity pueda recuperar la lectura correcta.
Reemplazo modular	Tornillos	Se aplicó como caso cerrado de reconstrucción modular y ahorro geométrico; otros fasteners se resolvieron mediante geometría original o proxies ligeros según su relevancia visual.

Nota. La modularización de fasteners integra datos, escena y optimización; no se limita a una agrupación visual.

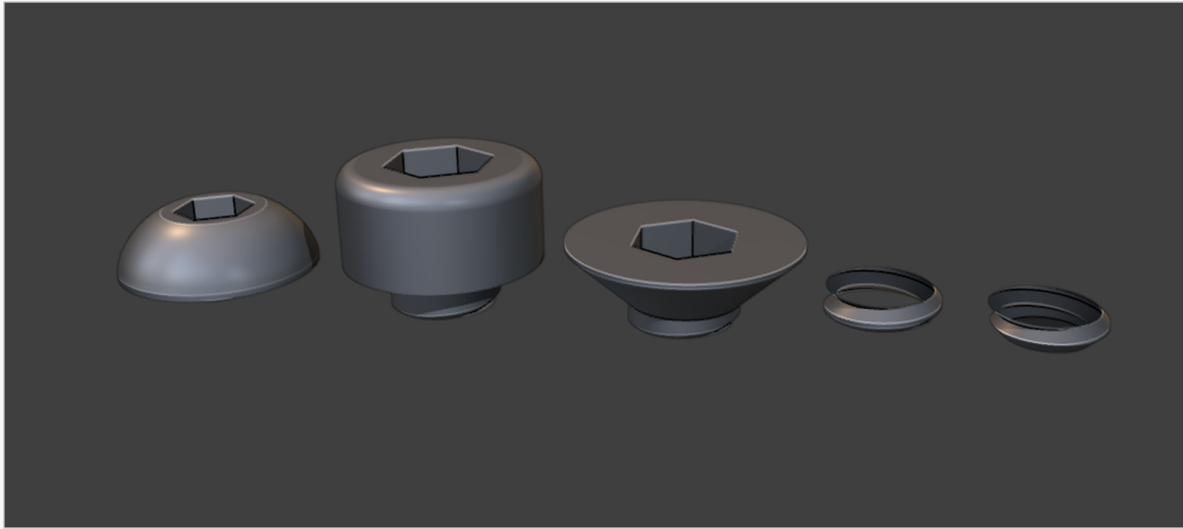
Elaboración propia a partir de la escena y del runtime de fasteners del proyecto.

La Figura 28 muestra las piezas base del sistema modular de tornillos. Estas cinco piezas permiten variar cabeza, diámetro, densidad vertical y longitud mediante repetición controlada del cuerpo. La Figura 29 muestra un ejemplo de tornillo generado con una cabeza y un arreglo de

siete segmentos de cuerpo.

Figura 28

Piezas base del sistema modular de tornillos.

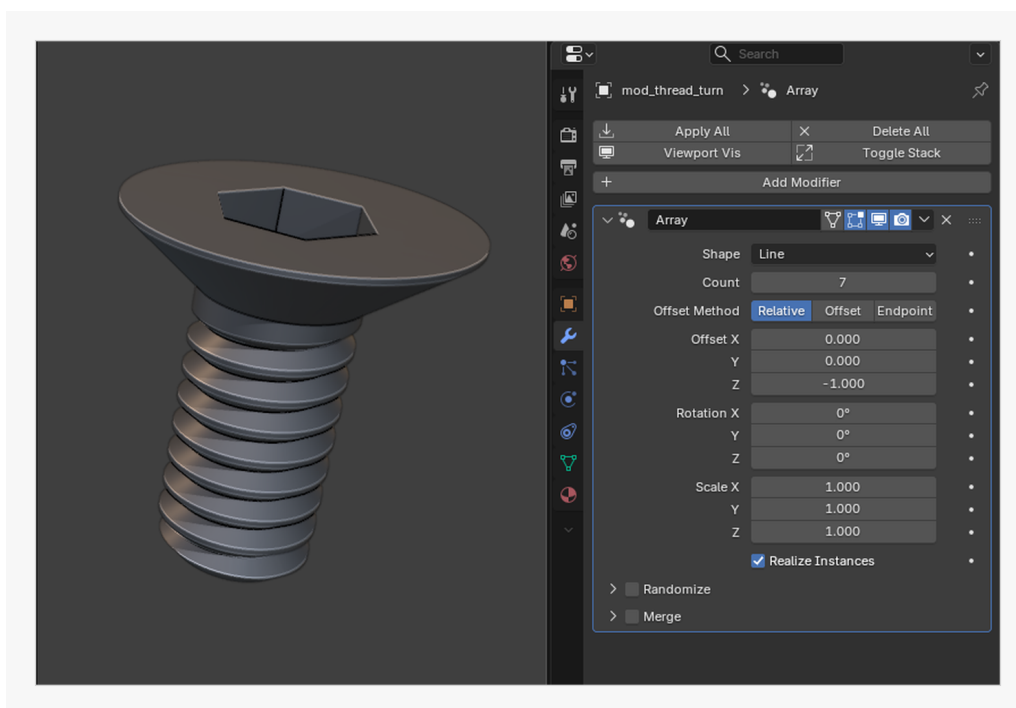


Nota. La reconstrucción modular de tornillos se basa en combinaciones de cabeza, diámetro y segmentos de cuerpo para ajustar longitud sin conservar una malla CAD única por cada tornillo. Elaboración propia.

En el flujo Blender–Unity, la escena runtime se exporta como unión de cuatro colecciones: masters bakeados, instancias del ensamblaje, masters primitivos de fasteners e instancias primitivas de fasteners. La decisión de exportar un master depende de su papel en el ensamblaje: puede ser solo referencia geométrica compartida o una ocurrencia visible que también debe aparecer en la escena final. El paquete de materiales previsto usa `BaseColor`, `Normal` y una máscara compacta $R=AO$, $G=Roughness$, $B=Curvature$, $A=Metallic$; la fibra de carbono se hornea desde el material construido en Blender. La asignación de fasteners a piezas madre se trata como revisión basada en manifest y no como inferencia automática cuando exista ambigüedad. El reemplazo modular queda limitado a tornillos; aplicar la misma reconstrucción a otros sujetadores se conserva como mejora futura del pipeline.

Figura 29

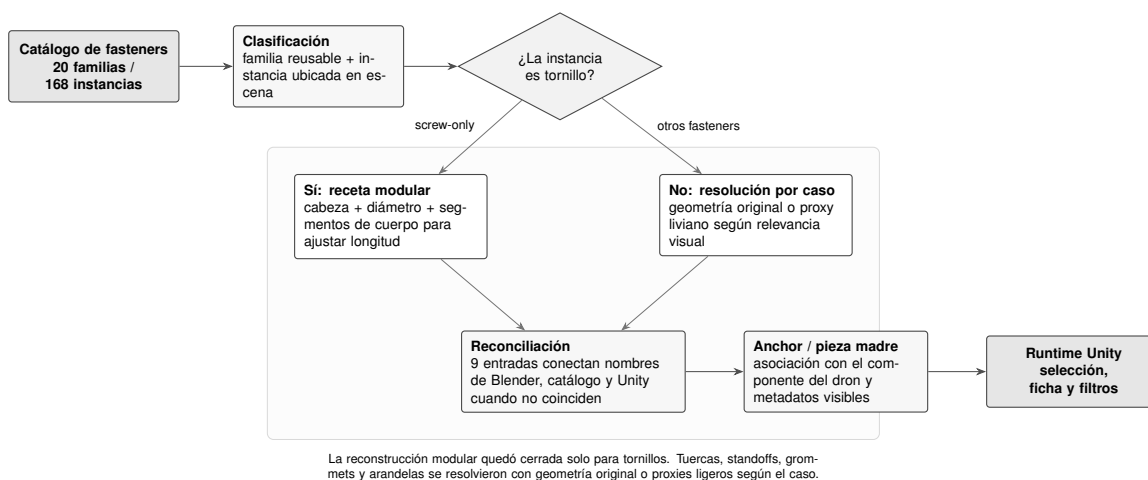
Ejemplo de tornillo generado mediante arreglo modular.



Nota. El ejemplo muestra un tornillo compuesto por una cabeza y siete segmentos repetidos de cuerpo, lo que ilustra el principio de variación de longitud mediante arreglo. Elaboración propia.

Figura 30

Sistema modular delimitado a tornillos y reconciliación de fasteners.



Nota. El diagrama distingue la reconstrucción modular aplicada solo a tornillos de la resolución por caso usada para otros fasteners, ya sea con geometría original o proxies ligeros. Elaboración propia.

La Figura 31 compara fasteners originales frente a proxies recortados generados con el addon y primitivas básicas. Esta evidencia visual no implica que todos los sujetadores hayan sido reconstruidos modularmente: documenta la coexistencia entre geometría original, proxies ligeros y reconstrucción modular aplicada solo a tornillos.

Figura 31

Fasteners originales y proxies ligeros usados en el pipeline.



Nota. La franja superior presenta fasteners en su forma original; la franja inferior muestra proxies ligeros generados con primitivas básicas y apoyo del addon. Elaboración propia.

Software especializado y costo del pipeline

La producción de activos combinó software libre, herramientas comerciales especializadas y addons de Blender. La Tabla 9 no busca fijar precios exactos, sino documentar las depen-

dencias del pipeline y su nivel relativo de costo, ya que las licencias varían por versión, región y plan.

Tabla 9*Software y addons del pipeline 3D*

Herramienta	Rol principal	Licencia	Rango	Interpretación
Blender	Modelado, limpieza, organización y exportación	Libre / gratuita	Sin costo directo	Núcleo ejecutado del pipeline de producción.
Unity	Integración, UI, shaders y despliegue web	Motor con plan de uso académico/personal	Sin costo directo en esta etapa	Punto de verdad del producto final.
MoI3D	Conversión CAD y control de salida poligonal	Comercial especializada	Medio / alto	Usada como apoyo para piezas donde la teselación requería mayor control.
STEPper	Importación de archivos STEP dentro de Blender	Addon especializado	Bajo / medio	Aportó una ruta directa para revisar, abrir o comparar piezas CAD sin salir del entorno de Blender.
RizomUV	Despliegue y revisión UV para texturizado	Comercial especializada	Medio / alto	Apoyó la preparación de UVs cuando la geometría requería control más fino para texturas y bake.
Hard Ops	Modelado hard-surface y booleans	Addon comercial	Bajo / medio	Complementó el remodelado y la reconstrucción de piezas.
Box Cutter	Corte rápido y booleans de precisión	Addon comercial	Bajo / medio	Aceleró iteraciones de remodelado y corrección.
Mesh Machine	Limpieza de bevels y tratamiento hard-surface	Addon comercial	Bajo / medio	Se evaluó y aprovechó para limpieza de superficies duras y resolución de detalles.
Marmoset Toolbag	Apoyo para bake/lookdev puntual	Comercial especializada	Medio / alto	Herramienta auxiliar del pipeline, no eje del runtime final.

Nota. “Rango” clasifica el tipo de inversión requerida sin fijar precios exactos sujetos a cambios comerciales. Elaboración propia a partir del pipeline del proyecto y de la investigación técnica asociada.

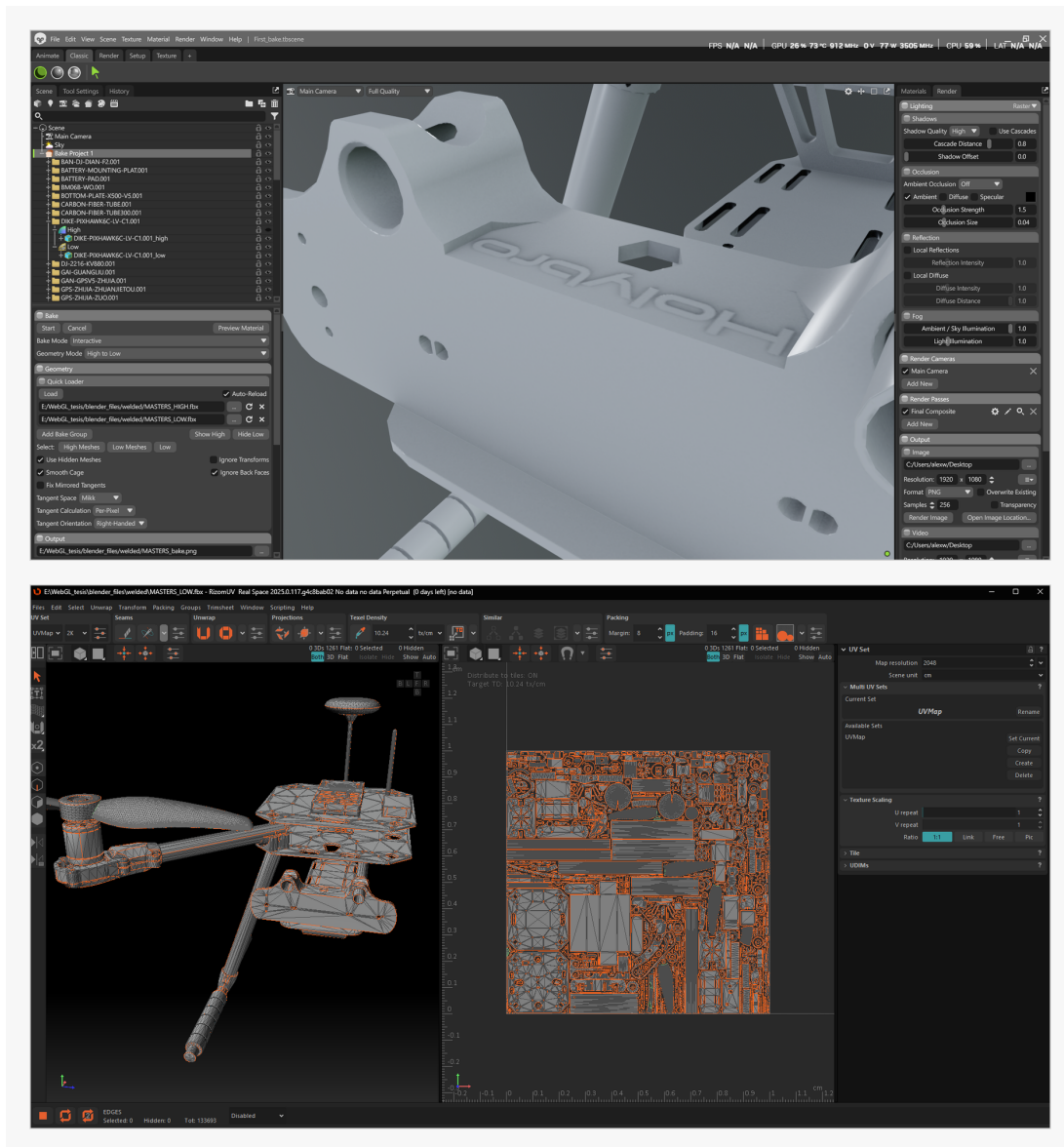
La Figura 32 documenta el uso de Marmoset Toolbag y RizomUV como herramientas auxiliares del pipeline de *baking*, *lookdev* y despliegue UV. Estas capturas se agrupan entre sí porque representan software especializado de preparación visual y texturizado, no estados de la app ni comparativas de importación CAD.

Automatización y addons creados para el proyecto

El volumen y la naturaleza de la geometría CAD hicieron inviable un flujo puramente manual. Por ello se desarrollaron y documentaron herramientas auxiliares para selección de piezas hermanas, instanciación, proxies, sustitución de mallas, preparación de bake y generación de manifests (cuyo resumen y funciones principales se describen en la Tabla 10 y se ilustran concep-

Figura 32

Software auxiliar para baking, lookdev y despliegue UV.



Nota. La captura superior corresponde a Marmoset Toolbag como apoyo de *baking* y revisión visual; la inferior corresponde a RizomUV como herramienta de despliegue y revisión UV. Elaboración propia.

tualmente en la Figura 33).

El *CAD Symmetry Instancer* se creó para detectar piezas repetidas, reconstruir relaciones de instancia y conservar transformaciones espaciales. Sus primeras versiones evidenciaron problemas propios de la geometría CAD importada: orígenes globales en el centro de la escena, rotaciones horneadas en vértices, simetrías radiales parciales y piezas cilíndricas donde técnicas como PCA podían producir giros ambiguos. Las versiones posteriores incorporaron estrategias más robustas, como validación por nombre, volumen, conteo de vértices, registro por matrices y variantes con ICP para casos de teselación irregular.

El *CAD Mesh Swapper* resolvió otro problema frecuente: reemplazar la geometría interna de una pieza dañada por una versión limpia sin alterar su posición global. La herramienta permitía inyectar una malla saneada en un objeto objetivo y propagar el cambio a las instancias que compartían el mismo dato de malla, conservando materiales y transformaciones. En paralelo, el *CAD Fastener Proxy Automator* se orientó a tornillería y repetitivos, analizando proporciones de cabezal y caña para generar proxies de baja complejidad con tipologías como socket, button o flat.

Finalmente, se desarrollaron scripts no destructivos para preparar el bake y la exportación final: creación de imágenes objetivo para mapas PBR, empaquetado de canales en máscaras runtime, y generación de un manifest de Blender con masters, instancias, roles, *transforms*, *bounds* y candidatos de pieza madre para fasteners. Estos scripts no sustituyeron la revisión manual, pero hicieron auditable el estado de la escena y redujeron el riesgo de exportar una jerarquía incompleta o ambigua.

La Figura 34 muestra el estado de alta resolución importado mediante STEPper y utilizado como fuente para el bake. Se presenta con vista sombreada y *wireframe* apilados para conser-

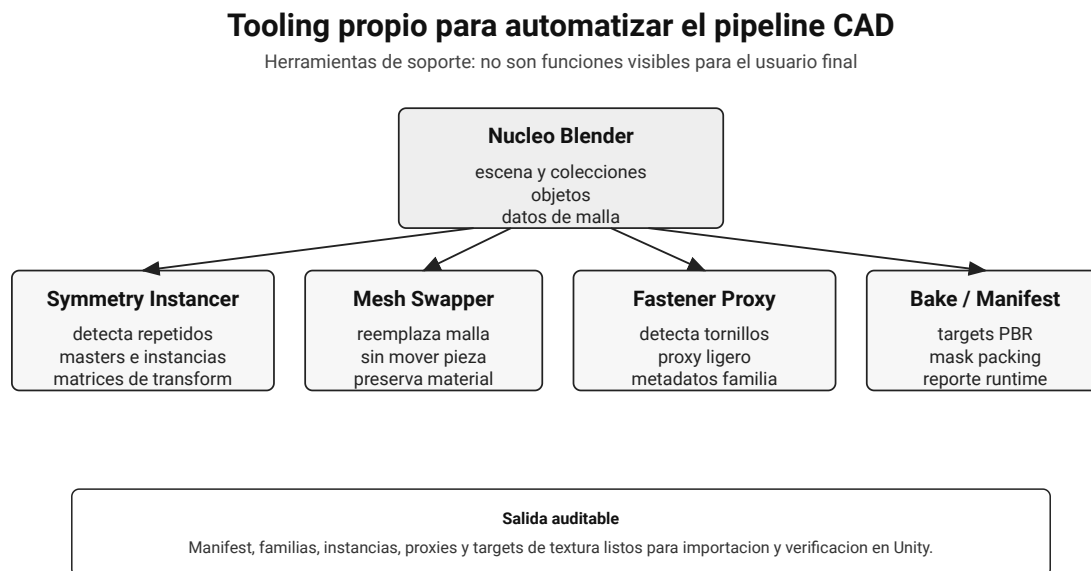
Tabla 10*Automatizaciones y addons asociados al pipeline*

Herramienta	Problema que resuelve	Salida técnica	Estado e interpretación
CAD Symmetry Instancer	Piezas CAD repetidas exportadas como mallas únicas o con transformaciones horneadas	Relación master/instancias conservando orientación y posición	Ejecutado e iterado; fue central para reducir trabajo manual sobre familias repetidas.
Variante MoI3D ICP	Piezas con teselación irregular y conteo de vértices inconsistente	Alineación por PCA/ICP para recuperar una instanciación posible	Evaluado y documentado; aporta una ruta para geometría CAD que no admite correspondencia topológica directa.
CAD Mesh Swapper	Mallas dañadas o mal convertidas que requieren reemplazo sin mover la pieza	Sustitución de geometría local limpia preservando transform global y materiales	Ejecutado de forma puntual; permitió corregir geometría sin romper el ensamblaje.
CAD Fastener Proxy Automator	Tornillería densa de bajo valor visual relativo en reposo	Proxies ligeros por tipología de cabezal y dimensiones estimadas	Ejecutado/evaluado; apoyó la estrategia de ahorro geométrico para repetitivos.
Bake Target Setup	Preparación repetitiva de imágenes y nodos para mapas PBR	Targets de Base-Color, Roughness, Metallic, Normal y AO	Documentado como apoyo no destructivo para bake en Blender.
Mask Packer	Exceso de texturas separadas para runtime	Máscara compacta $R=AO$, $G=Roughness$, $B=Curvature$, $A=Metallic$	Documentado para reducir archivos y simplificar consumo en Unity.
Runtime Manifest Exporter	Riesgo de perder masters, instancias o parentage de fasteners en exportación	Manifest con roles, <i>bounds</i> , transforms y candidatos de asociación	Ejecutado/documentado; hace auditable la escena antes de importarla en Unity.

Nota. La documentación de estas herramientas se conserva como parte del desarrollo porque su valor está en la automatización del pipeline, aunque no formen parte directa de la UI publicada. Elaboración propia a partir de `CAD_Symmetry_Instancer_Documentation.md`, `CAD_Bake_Automation_Workflow.md`, scripts Blender y documentos técnicos del proyecto.

Figura 33

Tooling propio para automatizar el pipeline CAD.



Nota. La figura resume el soporte técnico de automatización del pipeline; no representa funciones visibles para usuarios finales de la aplicación. Elaboración propia.

var el ancho de página sin perder legibilidad. Después, la Figura 35 muestra la malla optimizada usada como referencia del cierre geométrico, también con su lectura *wireframe*.

Evolución de la estrategia de LOD y migración técnica evaluada

La planificación inicial contemplaba generar LODs manuales desde Blender, exportarlos por jerarquía y configurarlos mediante grupos tradicionales. Esa estrategia sigue siendo válida conceptualmente: reducir triángulos en mallas lejanas disminuye trabajo de render y ayuda a controlar el presupuesto visual. Sin embargo, la investigación técnica del proyecto llevó a evaluar rutas nativas de *Mesh LOD* dentro del ecosistema Unity 6.x para reducir mantenimiento manual si el cierre del pipeline lo justifica.

Cabe precisar que la adopción plena de una ruta nativa de LOD no se documenta como cambio cerrado de la build. Lo defendible académicamente es explicarla como una decisión técnica evaluada: podría reducir la complejidad de mantenimiento frente a LODs manuales por pieza, alinear mejor el reimport final con un workflow nativo de motor y minimizar retrabajo si el cierre del pipeline exige nuevas iteraciones sobre la geometría (véase la Figura 36).

UX/UI, diseño visual y enfoque *mobile-first*

El diseño de la interfaz no fue una capa visual independiente de la arquitectura, sino una capa de decisión con respaldo teórico y consecuencias directas sobre la comprensión del ensamblaje. La versión con sustento de diseño mejor fundamentado es la interfaz *mobile-first*. La aplicación en escritorio existe, funciona y se mantiene correcta para uso y demostración; sin embargo, no fue re-diseñada con el mismo nivel de rigor sistémico específico para *desktop*. Por limitaciones de tiempo, la versión de escritorio se conservó como una adaptación funcional del layout móvil (véase la Figura 38).

La interfaz móvil se apoyó en varios principios convergentes:

Figura 34

Modelo de alta resolución usado como fuente de bake.



Nota. La vista sombreada y su equivalente *wireframe* evidencian la densidad geométrica usada como fuente de bake frente al activo optimizado. Elaboración propia.

Figura 35

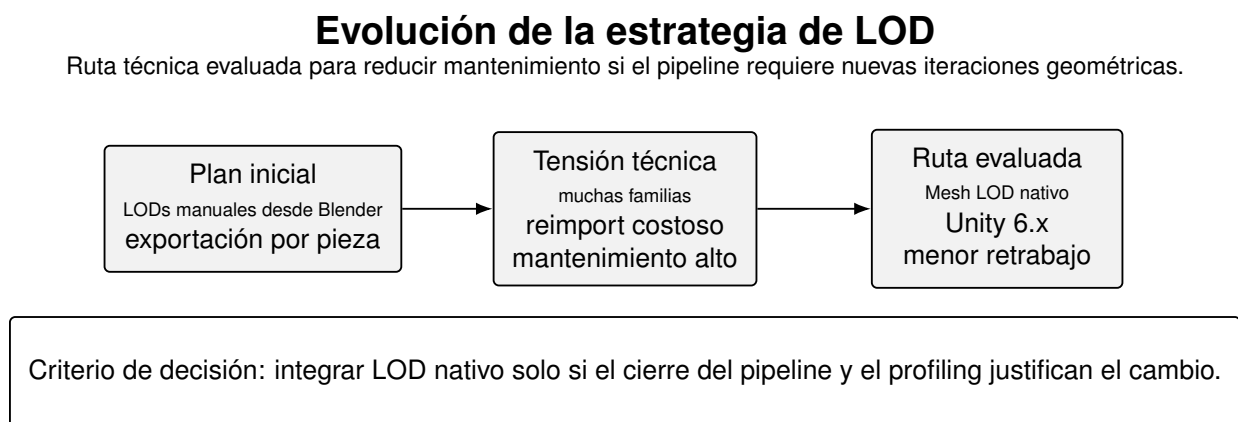
Malla optimizada del dron después del cierre geométrico.



Nota. La captura muestra el activo optimizado usado como referencia visual para el cierre del pipeline, con geometría reducida y lectura *wireframe* de la malla final. Elaboración propia.

Figura 36

Evolución de la estrategia de LOD: de planificación manual a ruta nativa evaluada.



Nota. La figura documenta una evolución de planeación técnica desde LODs manuales hacia una ruta nativa evaluada dentro del ecosistema Unity. Elaboración propia.

- jerarquía visual clara para separar navegación primaria, acciones de modo y detalle contextual (Norman, 2013);
- agrupación por proximidad, similitud y continuidad para evitar dispersión visual en paneles y tarjetas (Wertheimer, 1923; Köhler, 1929);
- soporte a la lectura técnica mediante color, contraste y consistencia visual (Ware, 2012);
- control de navegación y selección en entornos 3D con foco en orientación y reducción de fricción operativa (Bowman et al., 2004; Darken & Sibert, 1996).

En la práctica, estas decisiones se tradujeron en una barra inferior persistente de modos, un *bottom sheet* como contenedor principal de detalle, tarjetas de acción compactas y un flujo que minimiza los cambios bruscos de contexto (como se detalla en la Tabla 11 y se ilustra en las Figuras 37, 38 y 39).

Tabla 11*Criterios de diseño aplicados a la UI del prototipo*

Criterio	Aplicación en la app	Interpretación
Jerarquía visual	Separación entre Hero, navegación inferior, modos y ficha inferior.	Permite que el usuario identifique el flujo principal sin depender de menús complejos.
Diseño <i>mobile-first</i>	Barra inferior, <i>bottom sheet</i> , tarjetas táctiles y scroll elástico.	Es la capa de UI con respaldo teórico más consistente del proyecto.
Adaptación a escritorio	Reacomodo funcional del layout móvil sobre una pantalla más amplia.	Es utilizable para demostración y revisión técnica, pero no corresponde a un sistema de diseño de escritorio plenamente desarrollado.
Consistencia cromática y de estados	Paleta contenida, retroalimentación de selección y lectura por modos.	Refuerza la comprensión del sistema y evita ruido visual innecesario durante la inspección.

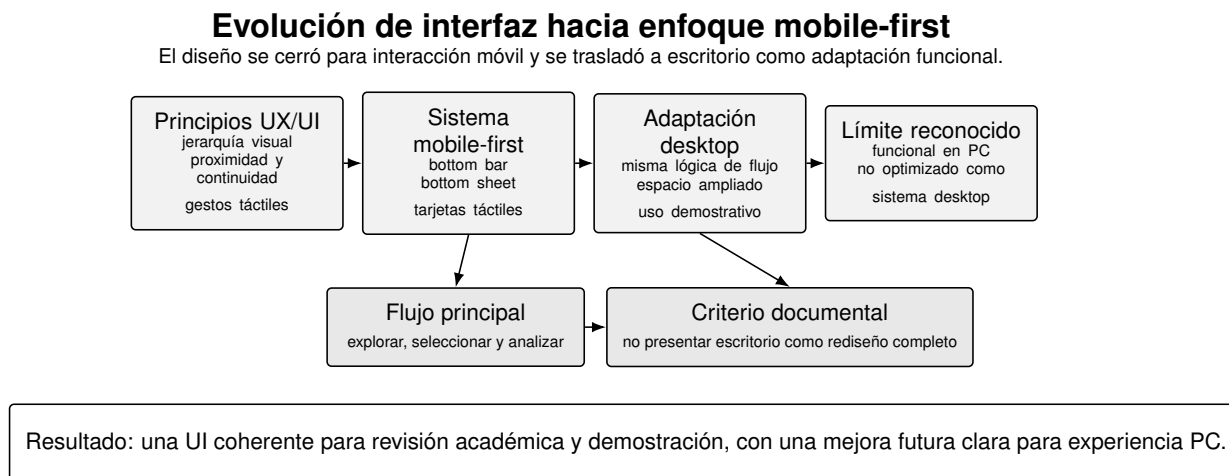
Nota. Esta tabla evita presentar la versión de escritorio como si hubiera recibido el mismo nivel de diseño sistémico específico que la interfaz móvil. Elaboración propia.

Ficha inferior, panel de información y jerarquía de selección

La ficha inferior no funciona como un panel lateral genérico ni como un contenedor decorativo. En la arquitectura de la build final, el *bottom sheet* es la superficie principal de información contextual. Su controlador asociado, `UIDetailsSheet`, sincroniza la selección activa

Figura 37

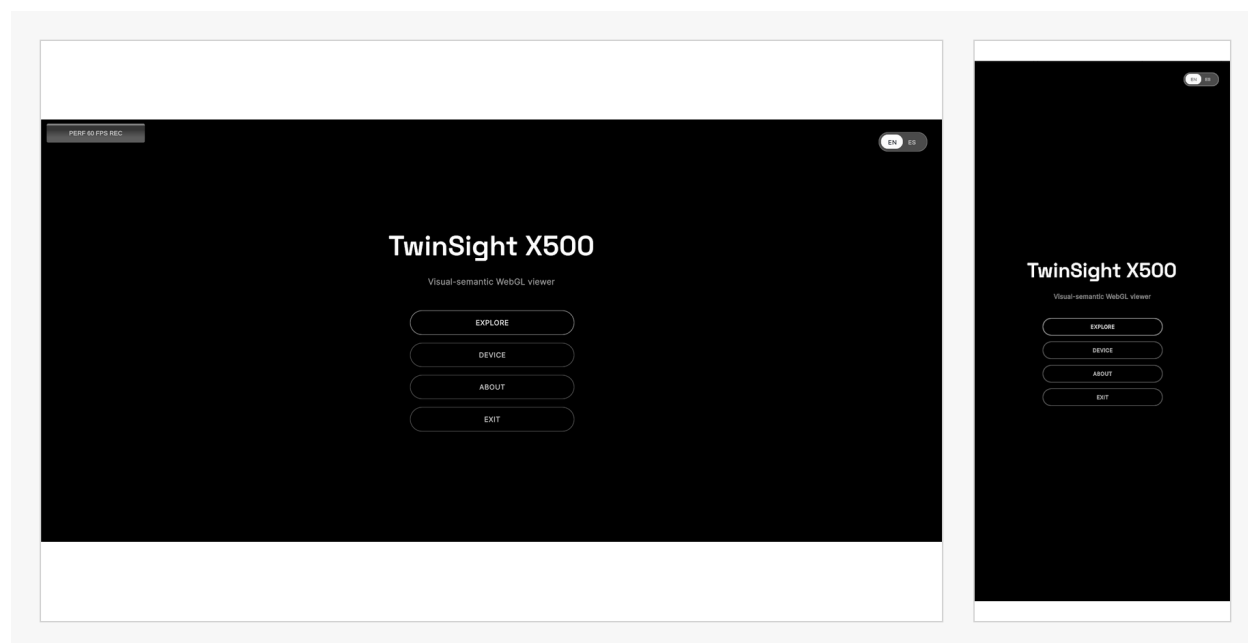
Evolución de la interfaz hacia el enfoque mobile-first.



Nota. Diagrama editorial de decisiones de interfaz que separa el diseño *mobile-first* de su adaptación funcional a escritorio. Elaboración propia.

Figura 38

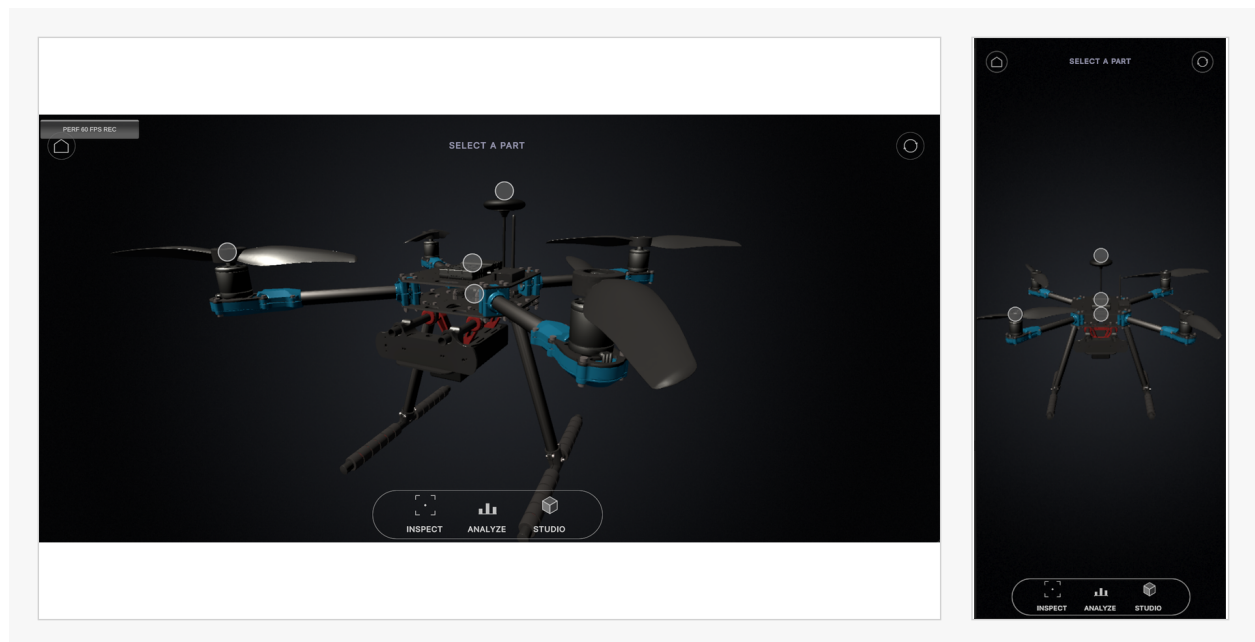
Hero móvil y adaptación funcional a escritorio.



Nota. La comparación muestra el Hero en escritorio junto a su versión móvil, evidenciando que la versión desktop opera como adaptación funcional del diseño *mobile-first*. Elaboración propia.

Figura 39

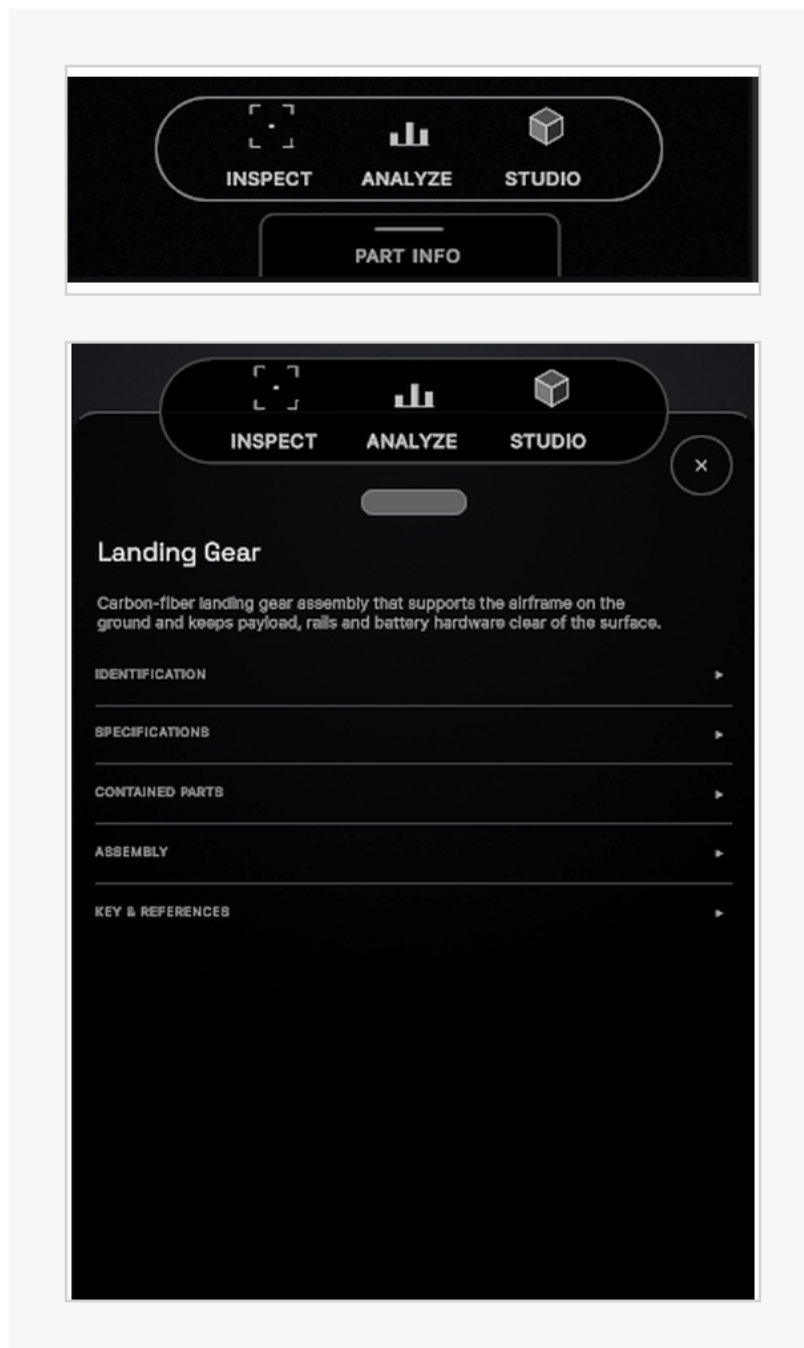
Explore móvil y adaptación funcional a escritorio.



Nota. La comparación ubica la vista de exploración dentro del canvas 3D de la app en escritorio y móvil, conservando el mismo flujo de navegación y lectura del modelo. Elaboración propia.

Figura 40

Acceso a información de pieza y panel contextual.



Nota. La lámina muestra la pestaña que abre el panel de información y el panel contextual de pieza; se presenta como acercamiento de interfaz, no como captura completa de pantalla móvil. Elaboración propia.

con el nombre visible de la pieza, la barra de previsualización, el foco de cámara y los bloques de información de identificación, especificaciones y ensamblaje.

Esta capa es relevante porque la selección del modelo no opera como una lista plana de objetos. El sistema puede resolver una interacción sobre cuatro niveles de lectura: **pieza madre**, cuando la selección corresponde al conjunto principal; **subpieza**, cuando el usuario entra a una geometría interna o nivel más fino del conjunto; **grupo de hotspot**, cuando la selección nace desde un marcador y representa una zona funcional; y **fastener**, cuando el objetivo es un sujetador individual o modular. En cada caso, la ficha inferior debe presentar la información coherente con el nivel seleccionado.

La misma jerarquía afecta otros subsistemas de la interfaz. El resaltado visual, el recentrado de cámara, el aislamiento y la apertura de la ficha deben compartir el mismo contexto semántico. De lo contrario, el usuario podría seleccionar una subpieza, pero recibir datos de una pieza madre, o activar un aislamiento que no corresponde al elemento descrito. Por esa razón, la ficha inferior se documenta como puente entre geometría, datos y comprensión técnica del ensamblaje.

Sistema de iconos procedurales y microinteracciones

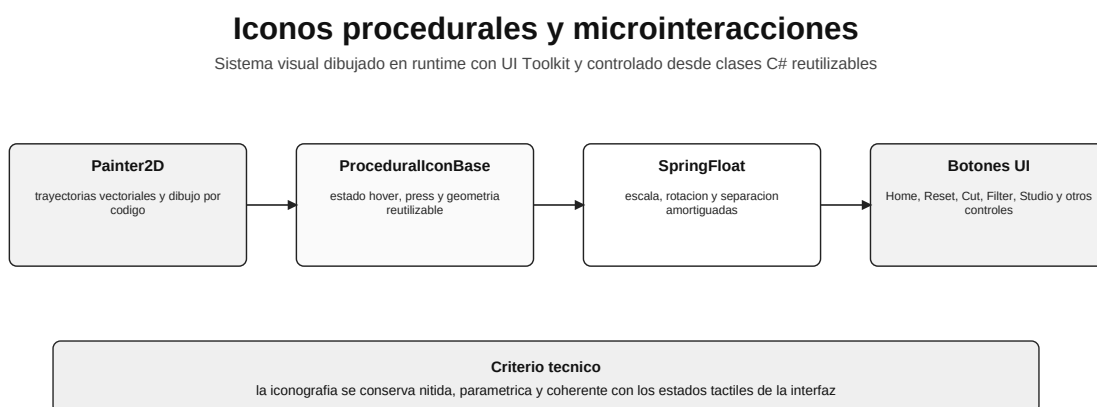
Una decisión distintiva del proyecto fue evitar un sistema de iconos basado en texturas o sprites tradicionales y reemplazarlo por iconografía procedural generada en código. La implementación actual usa `UI Toolkit` con `Painter2D` y `MeshGenerationContext` para dibujar trayectorias vectoriales y animarlas mediante una física ligera tipo resorte en `ProceduralIconBase`. Esta estrategia permitió conservar nitidez vectorial, evitar hojas de sprites y mantener control fino sobre *hover*, *press* y respiración visual de los botones, dentro del mismo sistema de UI runtime (Unity Technologies, 2024d).

En términos conceptuales, el sistema evolucionó desde un plan inicial de iconografía SVG

hacia una suite procedural escrita en C#. Los iconos de Home, Reset, *Inspect*, Cut, Explode, Filter, Isolate, Power, Shaders, *Studio* y otros botones ya existen como clases independientes, y su microinteracción se apoya en desplazamientos, escalados, rotaciones o separaciones controladas por una función `SpringFloat`. En consecuencia, el trabajo visual del proyecto no se limita a elegir íconos: incorpora geometría procedural, interpolación amortiguada y coherencia entre metáfora visual y función interactiva. La Figura 41 resume esta cadena entre dibujo vectorial, clase base procedural, microinteracción y botones visibles.

Figura 41

Sistema de iconos procedurales y microinteracciones implementadas en código.



Nota. El diagrama resume la arquitectura de implementación de los iconos procedurales y sus estados interactivos dentro de UI Toolkit. Elaboración propia.

Onboarding procedural y ayuda contextual

Además del sistema de iconos, el proyecto incorporó un subsistema de onboarding de primer uso para explicar controles, menús y microinteracciones sin depender de medios prerenderizados. La necesidad surgió porque la app no se limita a navegación 3D básica: combina selección jerárquica, *bottom sheet*, menús de inspección y análisis, sliders, modos visuales y variaciones

entre escritorio y móvil. En una primera lectura, esa densidad de interacción podía elevar la fricción de entrada si el usuario no recibía una guía integrada al propio producto.

La ruta inicialmente considerada consistía en usar GIFs o clips por tarjeta. Sin embargo, esa opción agregaba peso, duplicaba mantenimiento y se desacoplaba con facilidad del estado real de la interfaz. La solución ejecutada fue un renderer procedural en runtime, implementado sobre `Painter2D` mediante `OnboardingAnimationView`, sincronizado por `OnboardingController` con `copy`, *cues*, cambio de plataforma y estado del overlay. En lugar de reproducir video, cada tarjeta dibuja actor de interacción, objetivo y respuesta del sistema usando el mismo lenguaje visual del resto de la UI.

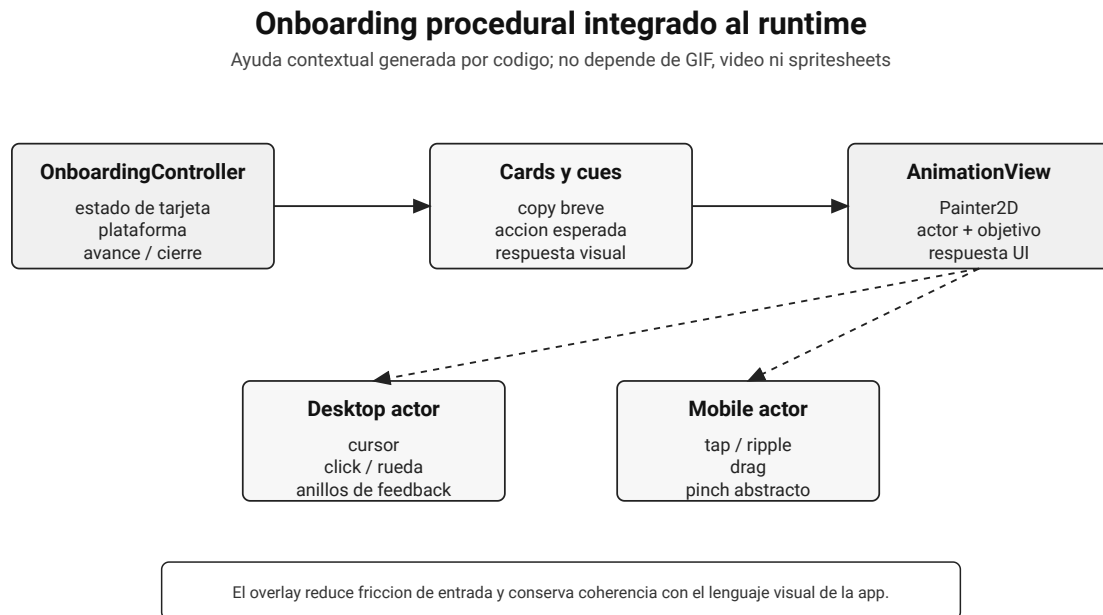
La versión actual cubre **15 tarjetas** y dos variantes de actor: **desktop** con cursor, botones de mouse, rueda y anillos de click; y **mobile** con *tap*, *ripple*, *drag* y *pinch* abstractos. Esta arquitectura permitió explicar `Navigate`, `Select / Deselect`, `Part Info`, *Inspect*, *Analyze*, *Studio* y sus subacciones sin recurrir a spritesheets, SVG runtime o videos embebidos. En consecuencia, el onboarding opera como una extensión del sistema de UI procedural, no como un recurso externo agregado al final del proyecto. La Figura 42 sintetiza la relación entre controlador, tarjetas, vista procedural y variantes de actor por plataforma, mientras que las Figuras 43, 44, 45, 46 y 47 muestran las tarjetas visibles integradas en la build en grupos de tres.

Arquitectura operativa

La arquitectura de la aplicación se organiza en cuatro capas de abstracción para garantizar el desacoplamiento entre el diseño de interfaz y la simulación interactiva. La capa de Presentación gestiona el layout responsivo mediante UI Toolkit; la capa de Orquestación coordina la lógica global del flujo de interacción; la capa de Servicios de Escena implementa funciones específicas como corte, explosión y simulación térmica; y la capa de Datos contiene los assets y

Figura 42

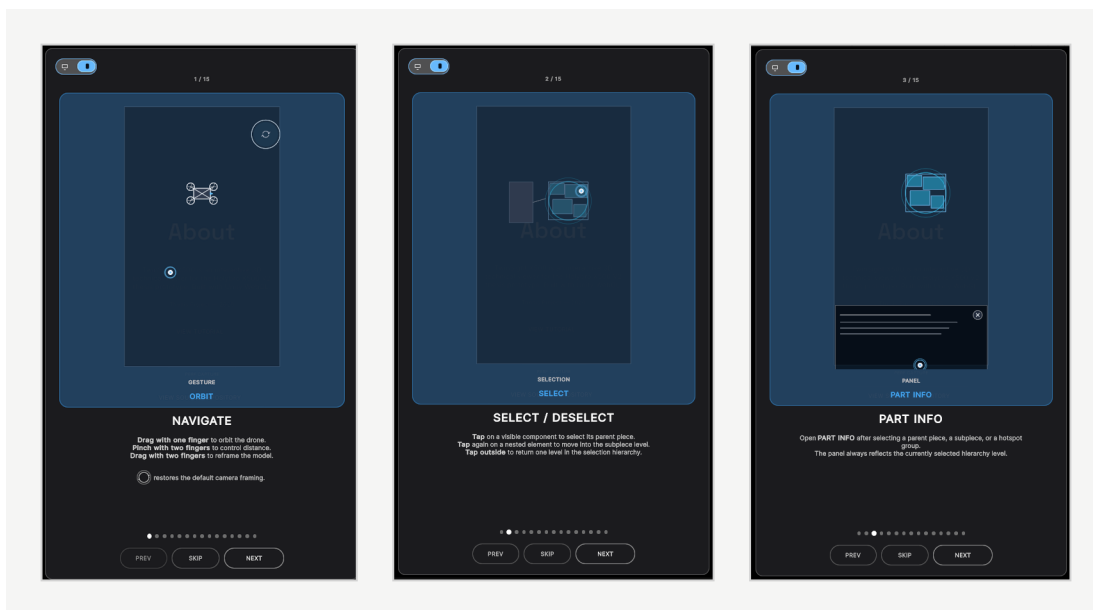
Onboarding procedural de primer uso integrado al runtime.



Nota. El diagrama resume la arquitectura de ayuda contextual generada por código; no depende de GIF, video ni spritesheets. Elaboración propia.

Figura 43

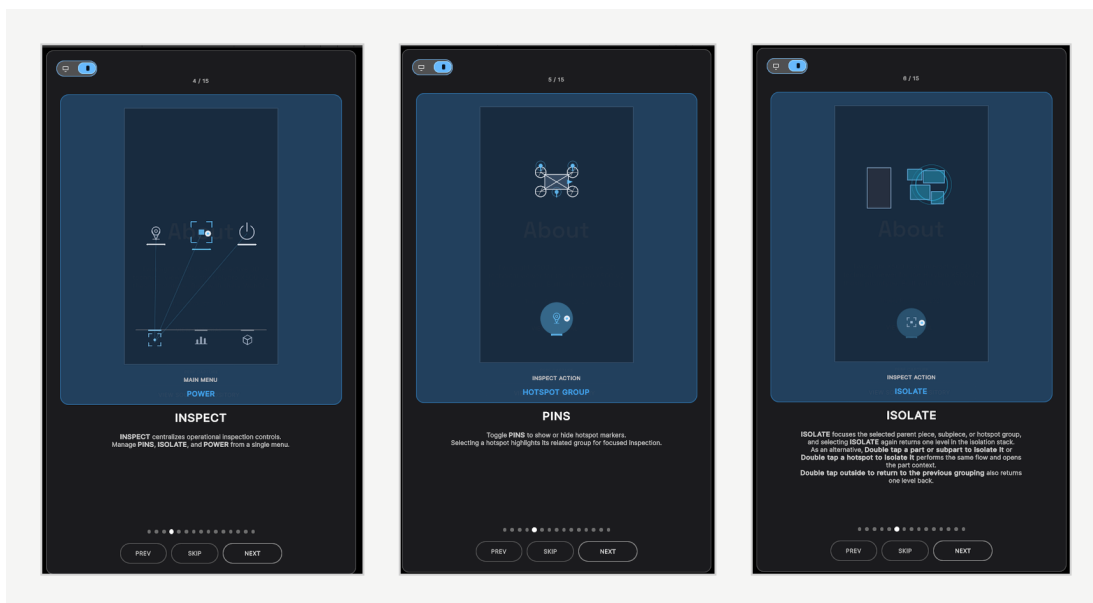
Tarjetas del onboarding procedural: navegación, selección e información.



Nota. De izquierda a derecha se presentan las tarjetas de navegación, selección/deselección e información de pieza. Elaboración propia.

Figura 44

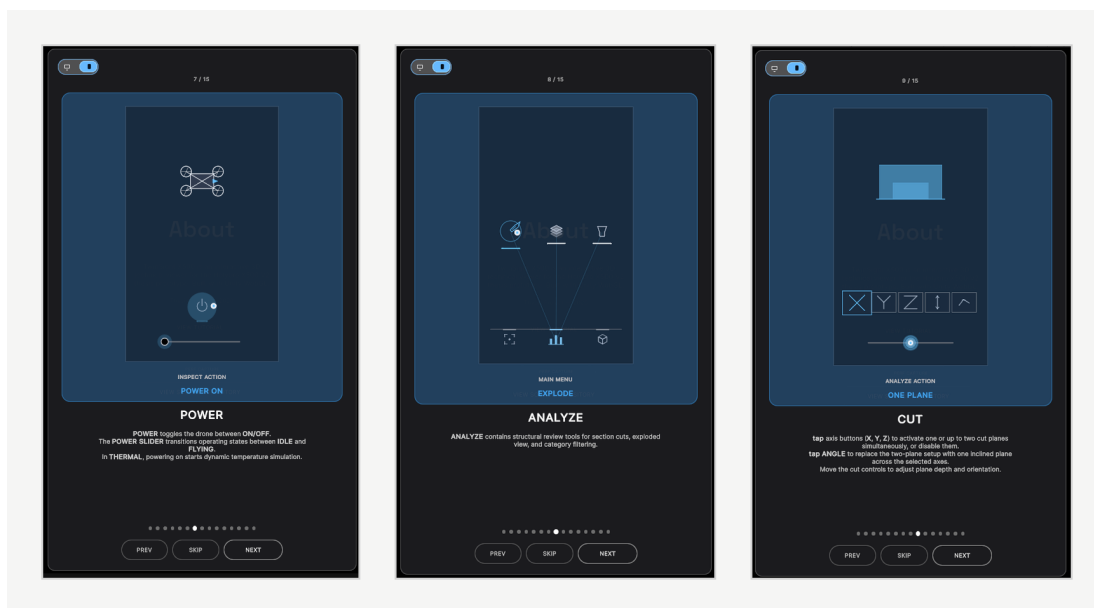
Tarjetas del onboarding procedural: inspección, hotspots y aislamiento.



Nota. De izquierda a derecha se presentan las tarjetas de *Inspect*, pines/hotspots y aislamiento. Elaboración propia.

Figura 45

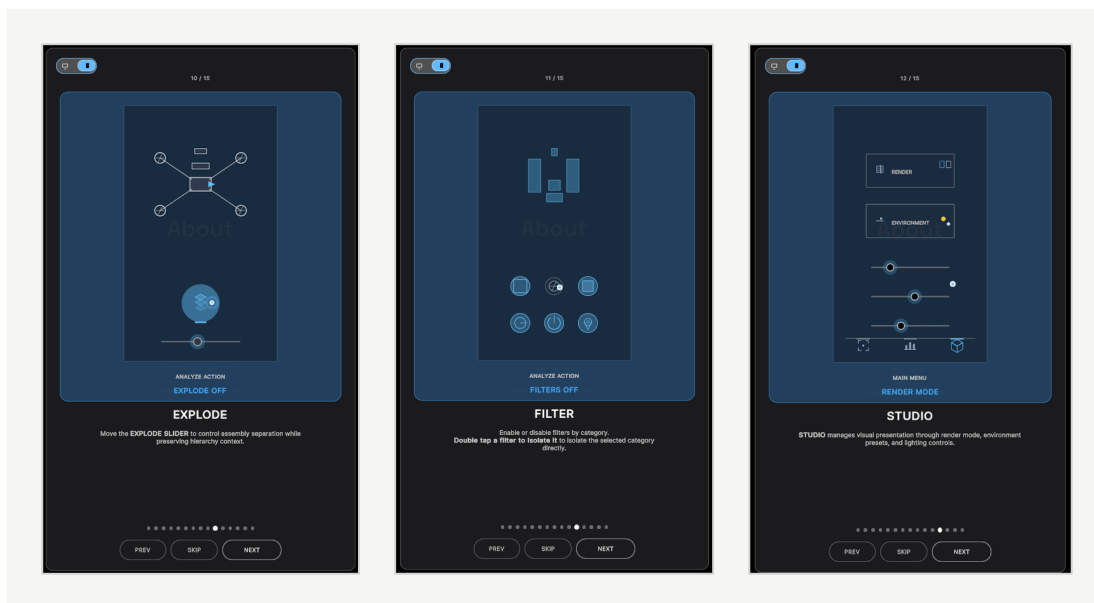
Tarjetas del onboarding procedural: energía, análisis y corte.



Nota. De izquierda a derecha se presentan las tarjetas de energía/carga, *Analyze* y corte transversal. Elaboración propia.

Figura 46

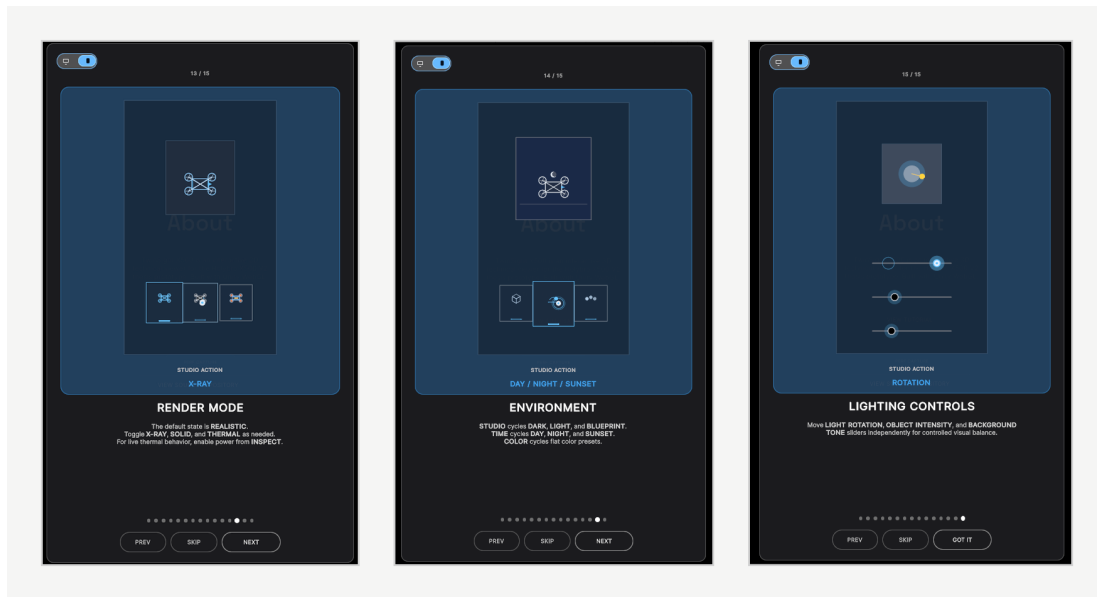
Tarjetas del onboarding procedural: explosión, filtros y Studio.



Nota. De izquierda a derecha se presentan las tarjetas de vista explosionada, filtros y *Studio*. Elaboración propia.

Figura 47

Tarjetas del onboarding procedural: modo de render, ambiente e iluminación.



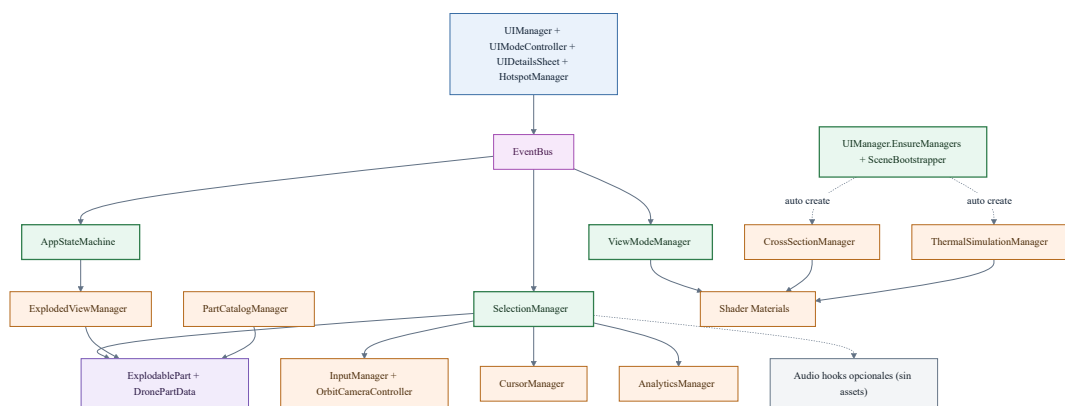
Nota. De izquierda a derecha se presentan las tarjetas de modo de render, entorno e iluminación. Elaboración propia.

catálogos serializados del dron (véase la Figura 48).

Para evitar el acoplamiento rígido entre controladores de interfaz y managers de escena, el sistema implementa un bus de eventos (*EventBus*) bajo el patrón de publicación/suscripción. De este modo, eventos como la selección de una pieza o el cambio de modo de renderizado se notifican al bus sin que el emisor conozca la implementación de los receptores, lo que simplifica la extensión del visor (véase la Figura 49).

Figura 48

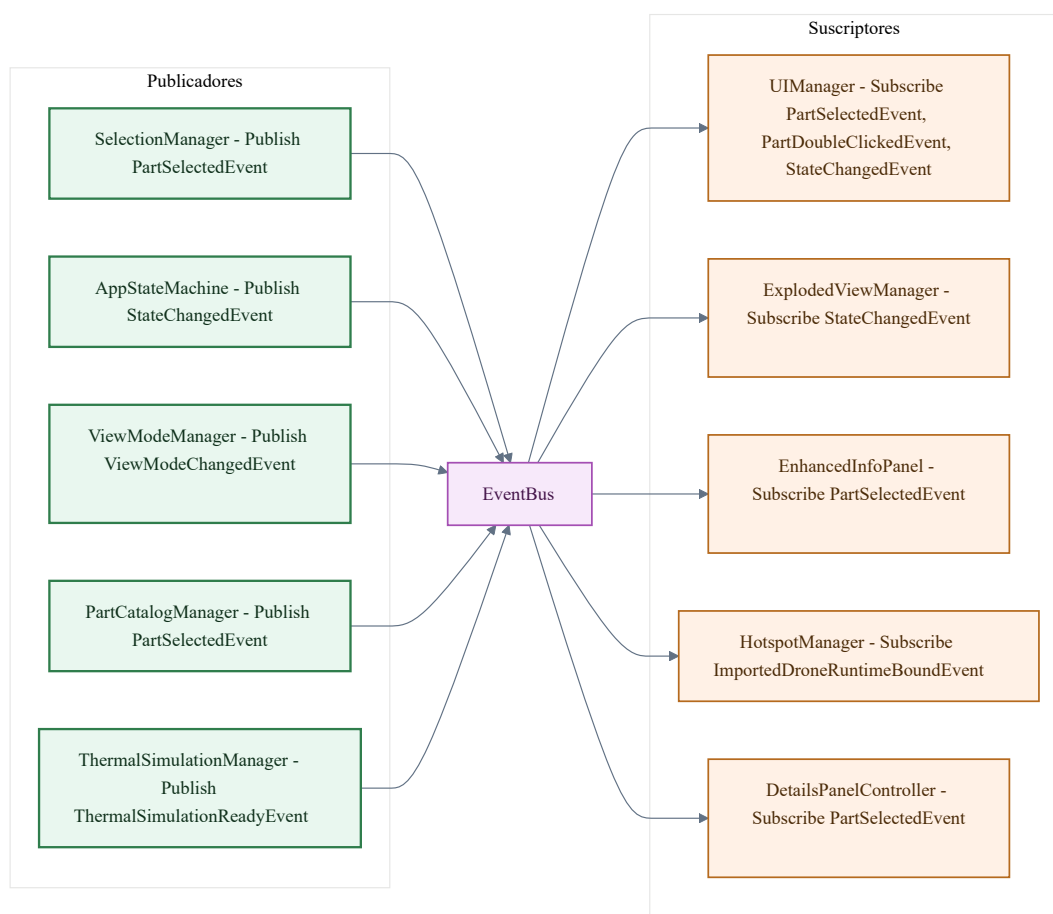
Arquitectura por capas del sistema.



Nota. Organización de la aplicación en cuatro capas independientes: presentación (UI), orquestación (manager general y estados), servicios de escena (cámara, corte, explosión y térmico) y datos. Elaboración propia.

Figura 49

Flujo de comunicación EventBus bajo el patrón publish/subscribe.

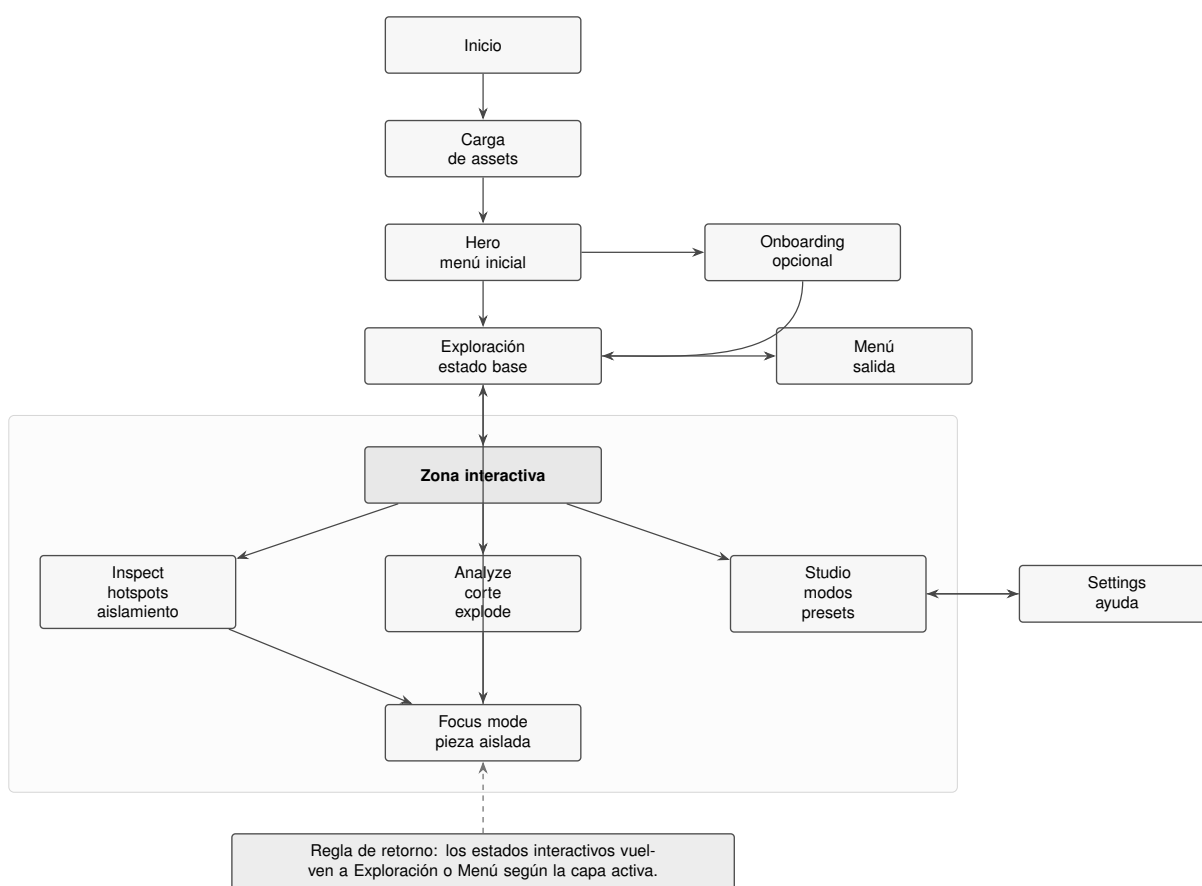


Nota. Diagrama del flujo de mensajes desacoplado entre emisores (UI y managers) y suscriptores de eventos mediante el bus central. Elaboración propia.

La secuencia de vistas y el ciclo de vida interactivo están regidos por una máquina de estados finita (AppStateMachine). Esta máquina organiza las transiciones del usuario desde la pantalla inicial, pasando por el onboarding guiado, hasta la exploración libre del modelo 3D y la activación de herramientas analíticas o presets de sombreado (véase la Figura 50).

Figura 50

Máquina de estados de la aplicación (AppStateMachine).



Nota. Máquina de estados finitos que controla los estados del visor y las transiciones válidas durante la sesión del usuario. Elaboración propia.

Bootstrap y saneamiento del import

La escena final depende de dos mecanismos complementarios.

Garantía de managers. Durante la inicialización, `UIManager` asegura la presencia de los managers mínimos requeridos por el visor: selección, explode, catalogación de runtime, notificaciones, hotspots, clipping, visibilidad, estado del dron y subsistema térmico.

Reparación del modelo importado. `ImportedDroneRuntimeBinder` corrige la jerarquía del dron importado en runtime. Su trabajo incluye:

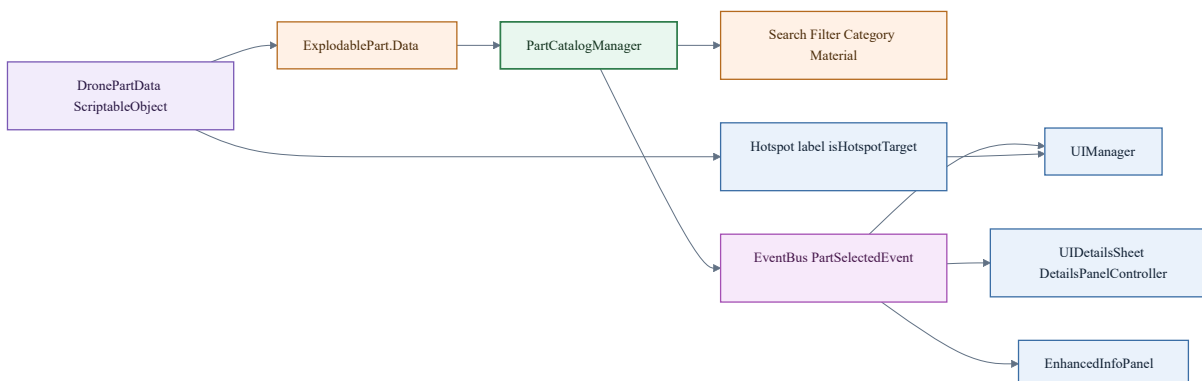
- resolver anchors por id;
- reubicar objetos huérfanos de nivel superior o anidados;
- sintetizar grupos para fasteners y misceláneos;
- reconstruir cachés de `ExplodedViewManager`, `PartVisibilityManager`, `PartCatalogManager`, `CrossSectionManager`, `ViewModeManager` y el subsistema térmico.

Este saneamiento es decisivo porque traduce la taxonomía académica del proyecto a una escena navegable por la app. Sin esa reparación, la coherencia entre selección, *bottom sheet*, categorías, vistas analíticas y estado térmico se rompería.

La Figura 51 resume el tránsito de `DronePartData` desde el asset o catálogo hasta los controladores de interfaz y los servicios de escena.

Figura 51

Flujo de datos DronePartData desde asset y catálogo hasta la interfaz.



Nota. La figura resume la ruta de normalización y consumo de DronePartData en runtime, desde los activos importados y el catálogo hasta los controladores visibles de interfaz y escena. Elaboración propia.

Flujo visible de la build final

El flujo principal de uso es:

Hero ->Explore ->selección ->bottom sheet ->Inspect / Analyze /
Studio

Hero. El Hero funciona como puerta de entrada al prototipo (véase la Figura 54). En la versión saneada para cierre, sus textos se anclan al caso real del Holybro X500 V2 y el acceso al repositorio queda cableado desde el submenú *About*. Las pantallas auxiliares del Hero, como *About*, selección de dispositivo y salida de la app, se documentan por separado en la Figura 55.

Onboarding y primer uso. En el primer ingreso, y también desde el botón *HELP* del Hero, la app puede abrir un onboarding guiado que resume gestos, selección, paneles, menús, sliders y modos visuales (véase la Figura 42). Esta capa cumple una función doble: reduce fricción

en el arranque y al mismo tiempo hace visible que la interfaz fue pensada como producto interactivo, no solo como demostración técnica del modelo, la cual se apoya en el sistema de iconos procedurales y microinteracciones de la Figura 41.

Explore y selección. La vista *Explore* corresponde al canvas 3D principal del modelo (véase la Figura 39). Al seleccionar una pieza, el sistema resalta el objeto y abre una ficha inferior (*bottom sheet*) con información de identificación, especificaciones y ensamblaje (véase la Figura 40). La selección puede corresponder a una pieza madre, subpieza, grupo de hotspot o fastener; por ello la ficha no solo muestra datos, sino que aclara el nivel de lectura activo. Cuando la selección pertenece a la categoría *Fasteners*, la ficha usa *FastenerMetadata* para mostrar familia, métrica, longitud, material, fuente CAD y anchor canónico inferido. La secuencia de hotspot, selección y selección directa se muestra en la Figura 59; los tres niveles de aislamiento se presentan en la Figura 60.

Inspect. Reúne hotspots, aislamiento de la pieza seleccionada y control de energía/carga del dron. Su menú principal forma parte de la barra inferior junto a *Analyze* y *Studio* (véase la Figura 52), mientras que el control de energía se muestra como acercamiento de botón en la Figura 56.

Analyze. Integra corte transversal, vista explosionada y filtros por categorías funcionales. La taxonomía activa en esta vista incluye *SkeletonAirframe*, *PropulsionSystem*, *Avionics*, *SensorsComms*, *PowerDistribution* y *Fasteners*. La Figura 57 muestra los controles de *Explode* y filtros, mientras que la Figura 61 muestra el efecto visible de *Explode*, *Cut* y *Filter* sobre el modelo.

Studio. Agrupa modos visuales, presets de presentación y control de entorno. El estado base es *Realistic*; la UI visible expone como tarjetas de shader *X-Ray*, *Solid View* o *Solid Color* y

Thermal. Además, el ciclo de presets de *Studio* permite activar *Studio*, *Studio Light* y *Blueprint*, junto con controles de ambiente, color y sliders de luz. El panel de *Studio* se presenta como menú de control en la Figura 58; los presets visuales se documentan más adelante en las Figuras 65 y 66.

La separación entre menús, controles y efectos evita mezclar capturas con funciones distintas. Los menús principales de la barra inferior se muestran en la Figura 52; las pantallas auxiliares del Hero en la Figura 55; los controles de *Power*, *Cut*, *Explode*, *Filter* y *Studio* en las Figuras 56, 57 y 58; y los efectos directos sobre el modelo en la Figura 61.

Figura 52

Menús principales de la barra inferior de la app.



Nota. De izquierda a derecha se presentan los menús principales *Inspect*, *Analyze* y *Studio*. Elaboración propia.

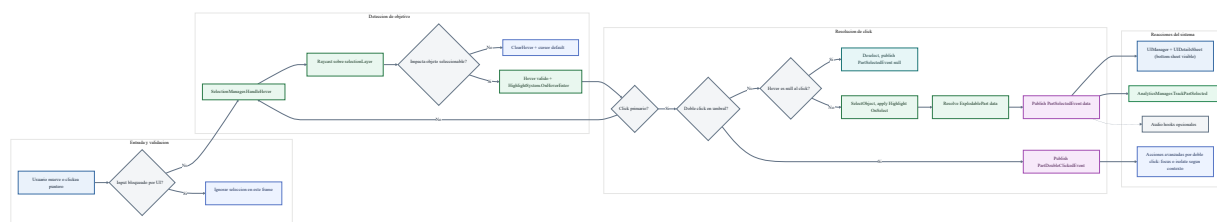
Los modos internos no publicados, como *Wireframe* y *Ghosted*, quedan implementados

pero ocultos. Por esa razón, no se tratan como capturas del flujo final visible.

La Figura 53 muestra el recorrido de selección desde el input del usuario hasta la sincronización con el *bottom sheet*.

Figura 53

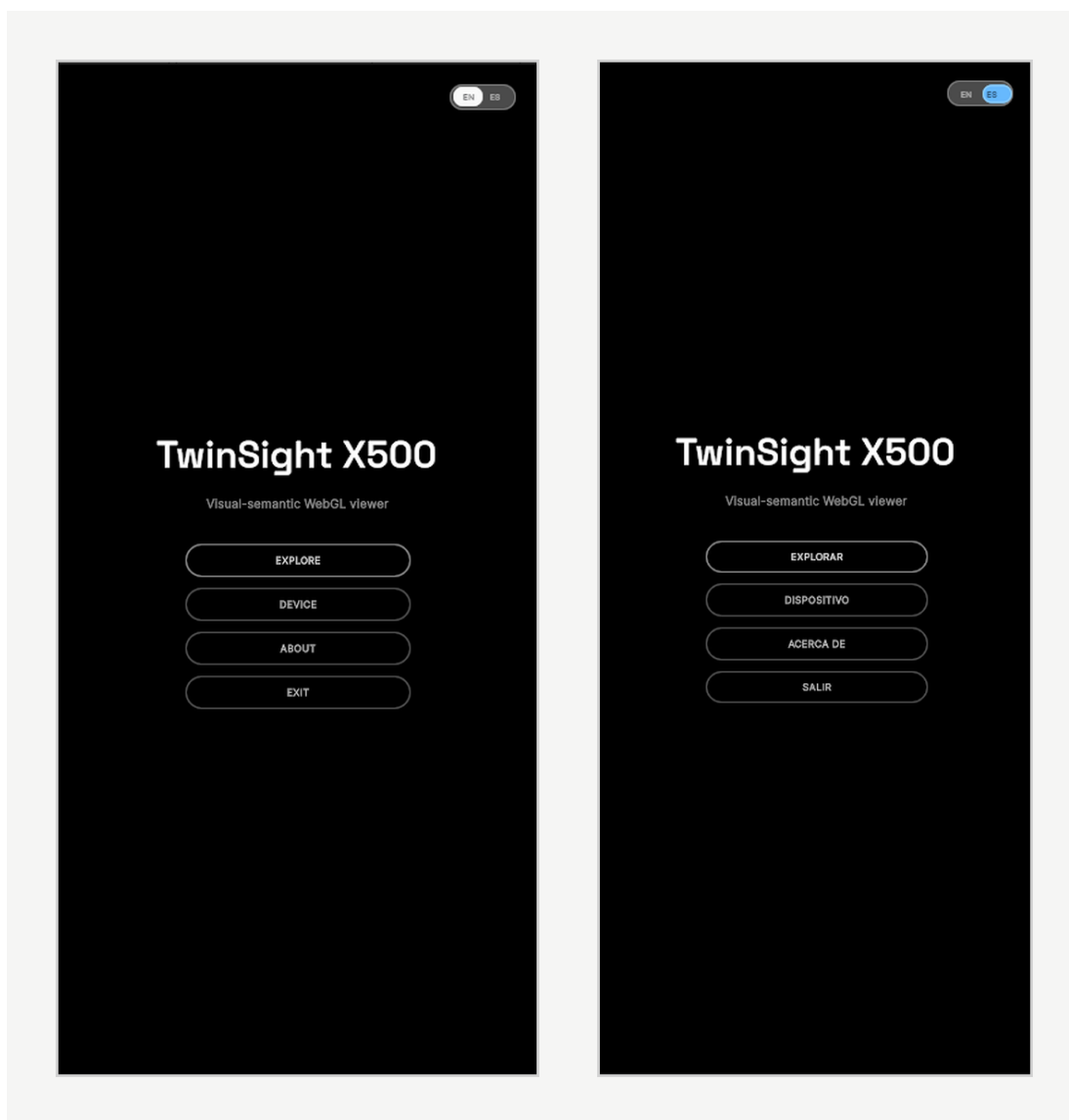
Flujo de selección de piezas desde input y raycast hasta la interfaz.



Nota. El diagrama sintetiza la cadena de selección visible: entrada del usuario, resolución de pieza o subpieza, eventos de escena, resaltado e información presentada en el *bottom sheet*. Elaboración propia.

Figura 54

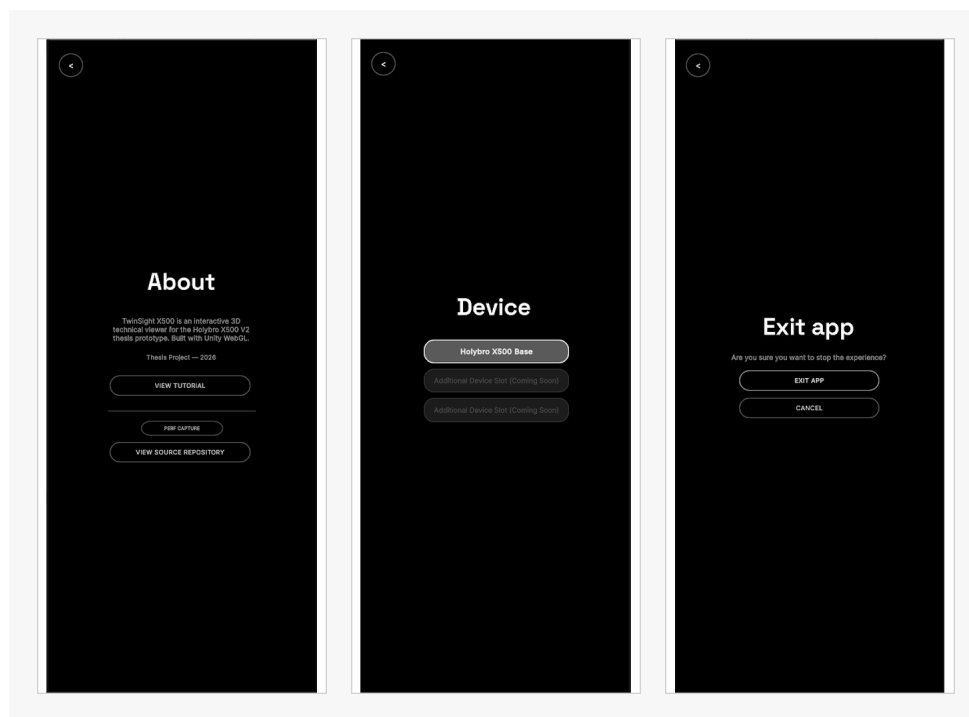
Hero móvil del visor en inglés y español.



Nota. De izquierda a derecha se presenta el Hero móvil en inglés y en español. Elaboración propia.

Figura 55

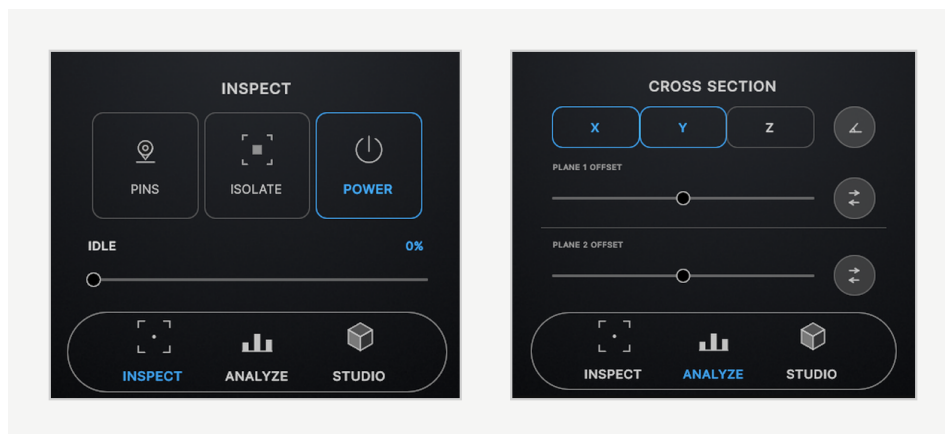
Pantallas auxiliares del Hero.



Nota. De izquierda a derecha se muestran las pantallas *About*, selección de dispositivo y salida de la app. Elaboración propia.

Figura 56

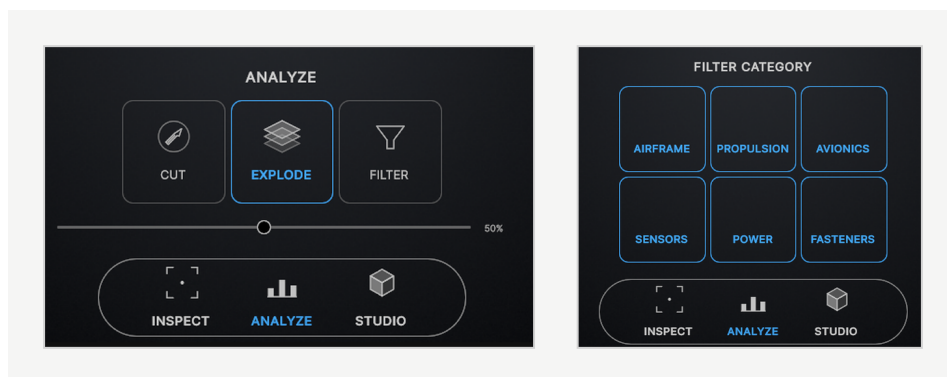
Controles de energía y corte transversal.



Nota. De izquierda a derecha se presentan los acercamientos de control de energía/carga y corte transversal. Elaboración propia.

Figura 57

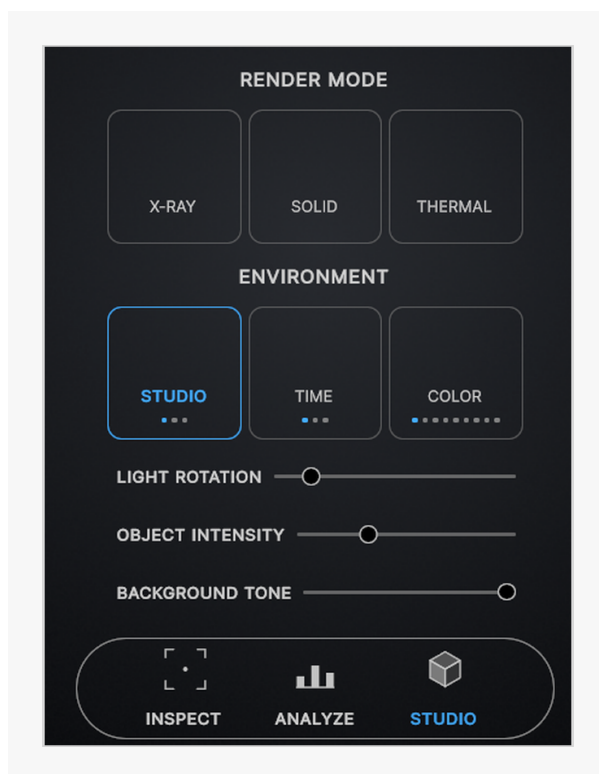
Controles de vista explosionada y filtros.



Nota. De izquierda a derecha se presentan los acercamientos de vista explosionada y filtros por categoría. Elaboración propia.

Figura 58

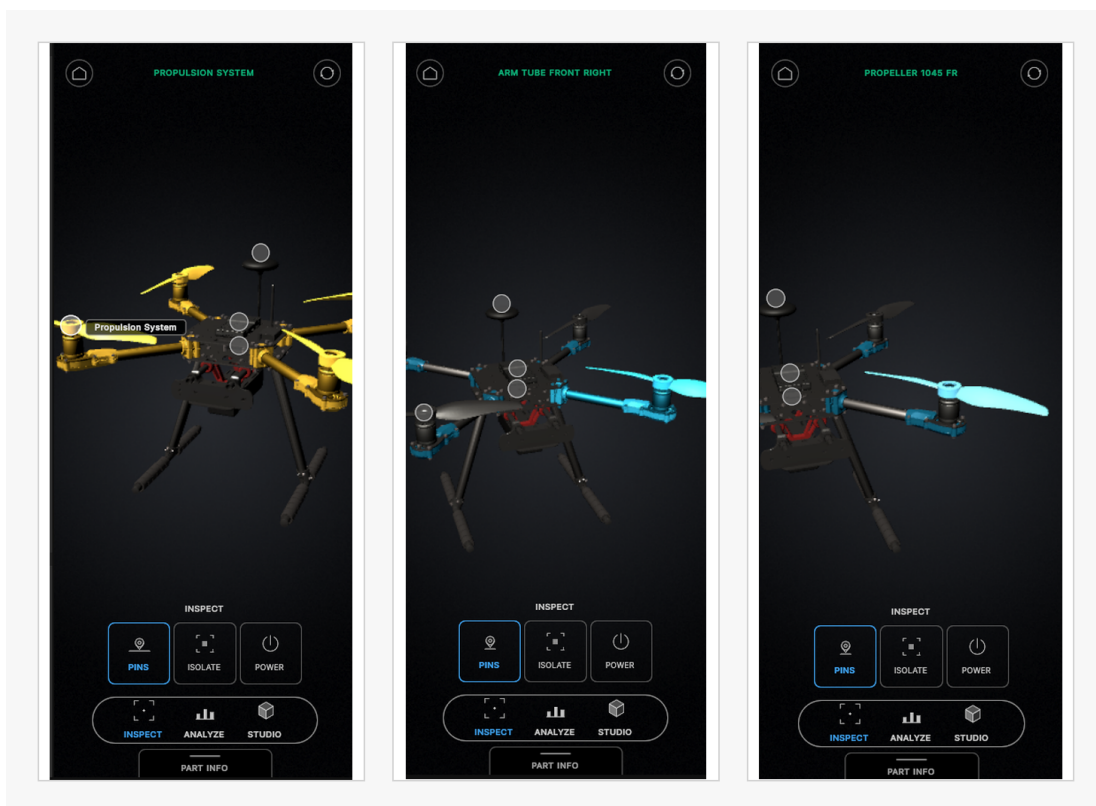
Panel de control del menú Studio.



Nota. La captura corresponde al menú de *Studio* como panel de controles; no representa el preset visual *Studio Dark*. Elaboración propia.

Figura 59

Hotspot y secuencia de selección en Explore.



Nota. De izquierda a derecha se presenta la selección de hotspot y dos estados de selección directa sobre piezas del modelo. Elaboración propia.

Figura 60

Secuencia de aislamiento por niveles en Inspect.



Nota. De izquierda a derecha se muestran tres niveles de aislamiento aplicados desde el flujo *Inspect*. Elaboración propia.

Figura 61

Efectos visibles de Explode, Cut y filtros sobre el modelo.



Nota. De izquierda a derecha se muestra la vista explosionada, el corte transversal y la aplicación de filtros por categoría. Elaboración propia.

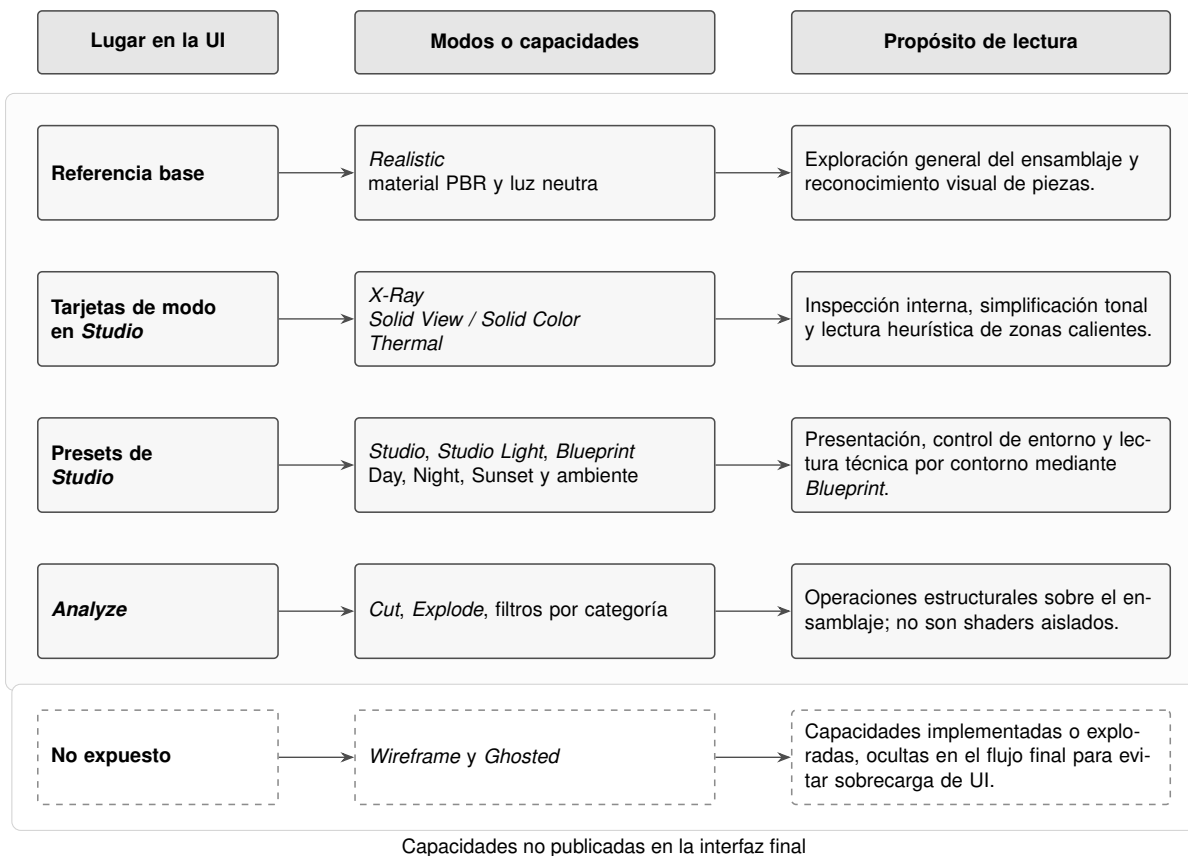
Sistema de visualización, shaders y entornos

`ViewModeManager` implementa siete modos de vista: *Realistic*, *XRay*, *Blueprint*, *SolidColor*, *Wireframe*, *Ghosted* y *Thermal*. La UI visible no los expone todos como tarjetas de shader: *X-Ray*, *Solid View/Solid Color* y *Thermal* aparecen como modos directos; *Blueprint* se publica mediante el preset de *Studio*; *Wireframe* y *Ghosted* permanecen como capacidades implementadas no visibles en el flujo final.

Para clasificar estas opciones de visualización según su funcionalidad técnica y su acceso en la interfaz, se definió un mapa de modos visuales por propósito (véase la Figura 62). Este mapa agrupa las representaciones en realistas, analíticas y de soporte de desarrollo, indicando en qué sección de la UI se expone cada una.

Figura 62

Mapa de modos visuales por propósito y lugar en la UI.



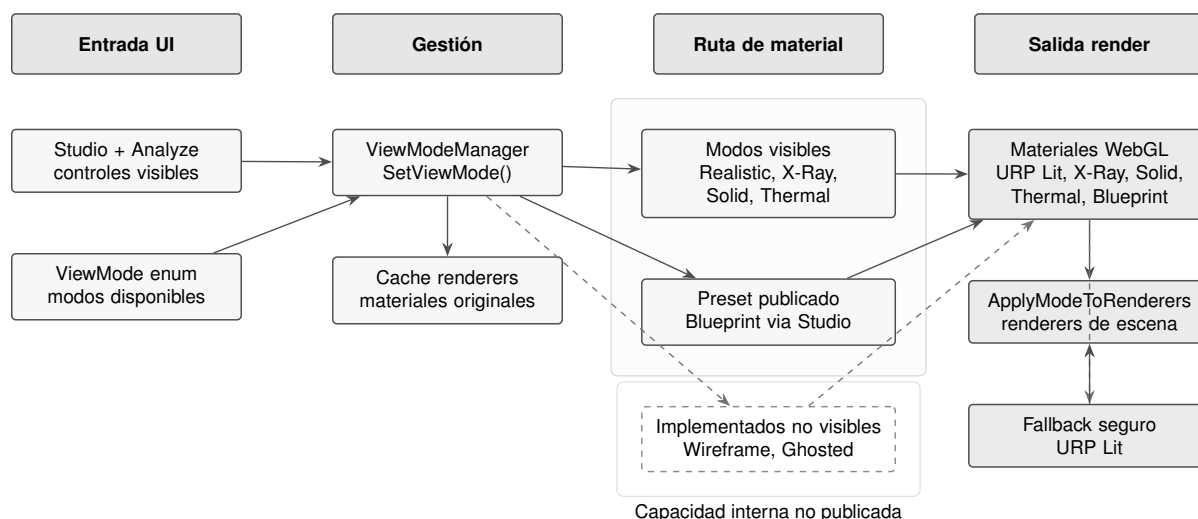
Nota. Las tarjetas de modo visibles se encuentran dentro de *Studio*: *X-Ray*, *Solid View* y *Thermal* funcionan como modos/shaders directos, mientras que *Studio*, *Studio Light*, *Blueprint* y los ambientes operan como presets de iluminación o apariencia. *Wireframe* y *Ghosted* permanecen como capacidades no publicadas en el flujo final. Elaboración propia.

A nivel de renderizado, cada modo interactúa de forma distinta con el pipeline gráfico y los materiales de soporte. La arquitectura del pipeline de shaders discrimina el flujo de datos según el modo de visualización activo, procesando pasadas específicas de contornos, transparencias

o gradientes térmicos (véase la Figura 63).

Figura 63

Pipeline de shaders por modo de visualización, distinguiendo entre modos implementados y modos expuestos en la UI final.



Nota. Diagrama del flujo de datos de renderizado y lógica de shaders aplicada por el motor para generar cada uno de los modos gráficos expuestos u ocultos. Elaboración propia.

Modo *Blueprint* y lectura analítica

El modo *Blueprint* merece una aclaración específica. La UI final no lo publica como tarjeta directa dentro del grupo de shaders, pero sí lo integra en el ciclo de presets de *Studio*. Al activarse desde esa tarjeta, el sistema cambia el modo base a *Blueprint* y aplica una lectura técnica basada en material propio, color de línea, color de fondo, umbral de borde, grilla y detección de contornos (véase la Figura 65). En términos de diseño visual, este modo se concibió para convertir la escena en una lectura de planos y siluetas técnicas sin perder la posibilidad de inspección 3D. Por ello funciona como puente entre un lenguaje de representación casi diagramático y el

modelo navegable en tiempo real.

Entornos, presets y color base

La capa de entorno combina dos ciclos. El primero organiza la lectura temporal de iluminación con presets *Day*, *Sunset* y *Night*. El segundo introduce presets de estudio y apariencia general, entre ellos *Studio*, *Studio Light* y *Blueprint*. Adicionalmente, la UI permite recorrer una base cromática compuesta por blanco, gris, negro, amarillo, naranja, verde, azul, púrpura y rojo. Esta capa fue útil para dos tareas: controlar el contraste visual del modelo y evaluar cómo la interfaz y los iconos procedurales responden a fondos claros u oscuros (como se resume en la Tabla 12, que detalla la relación de publicación de cada modo en la UI y su rol visual).

La evidencia visual se organiza en grupos pequeños para no mezclar controles con resultados. Los modos directos de render se muestran en la Figura 64; los presets de *Studio*, incluyendo *Blueprint*, se muestran en la Figura 65; y la variación de entorno/color se muestra en la Figura 66. Los efectos de *Explode*, *Cut* y *Filter* sobre el modelo se documentaron previamente en la Figura 61.

Modelo matemático operativo del subsistema térmico

El modo *Thermal* no implementa una simulación FEA ni una solución termodinámica de alta fidelidad. La versión final usa un solver reducido por componentes (*lumped-component model*) coherente con la arquitectura real del código: `DroneStateController` aporta el factor de carga, `ThermalSimulationManager` resuelve temperaturas por nodo y `ThermalViewController` traduce esas temperaturas a bandas visuales para shader y leyenda.

Para cada nodo térmico, la temperatura de equilibrio depende de la carga efectiva del

Tabla 12*Modos visuales, publicación en UI y lectura técnica*

Modo o preset	Estado UI	Rol visual	Interpretación
Realistic	Visible	Estado base PBR	Referencia principal del visor para lectura material y navegación general.
X-Ray	Visible	Inspección interna	Aporta transparencia y lectura estructural sin desmontaje completo.
Solid View / Solid Color	Visible	Color plano y contorno	Útil para lectura formal, evaluación de contraste y lectura de siluetas sin textura realista.
Thermal	Visible	Análisis visual guiado por temperatura heurística	Debe documentarse como simulación híbrida, no FEA.
Blueprint	Visible como preset de Studio	Lectura técnico-diagramática	Se activa desde el ciclo de Studio; funciona como preset técnico con líneas, grilla y contornos.
Wireframe	Oculto	Lectura de malla	Debe tratarse como capacidad implementada no publicada.
Ghosted	Oculto	Visualización analítica adicional	Se conserva como capacidad técnica no expuesta.
Day / Sunset / Night	Visible	Variación temporal de ambiente	Afectan contraste, percepción volumétrica y presentación.
Studio / Studio Light / Blueprint	Visible	Presets de estudio, iluminación y lectura técnica	Sirven para presentación, lectura técnica y control visual del modelo.

Nota. La tabla distingue entre tarjetas visibles, presets publicados indirectamente y capacidades ocultas, evitando confundir *Blueprint* con funciones no expuestas como *Wireframe* o *Ghosted*. Elaboración propia a partir del código y de la validación funcional del proyecto.

Figura 64

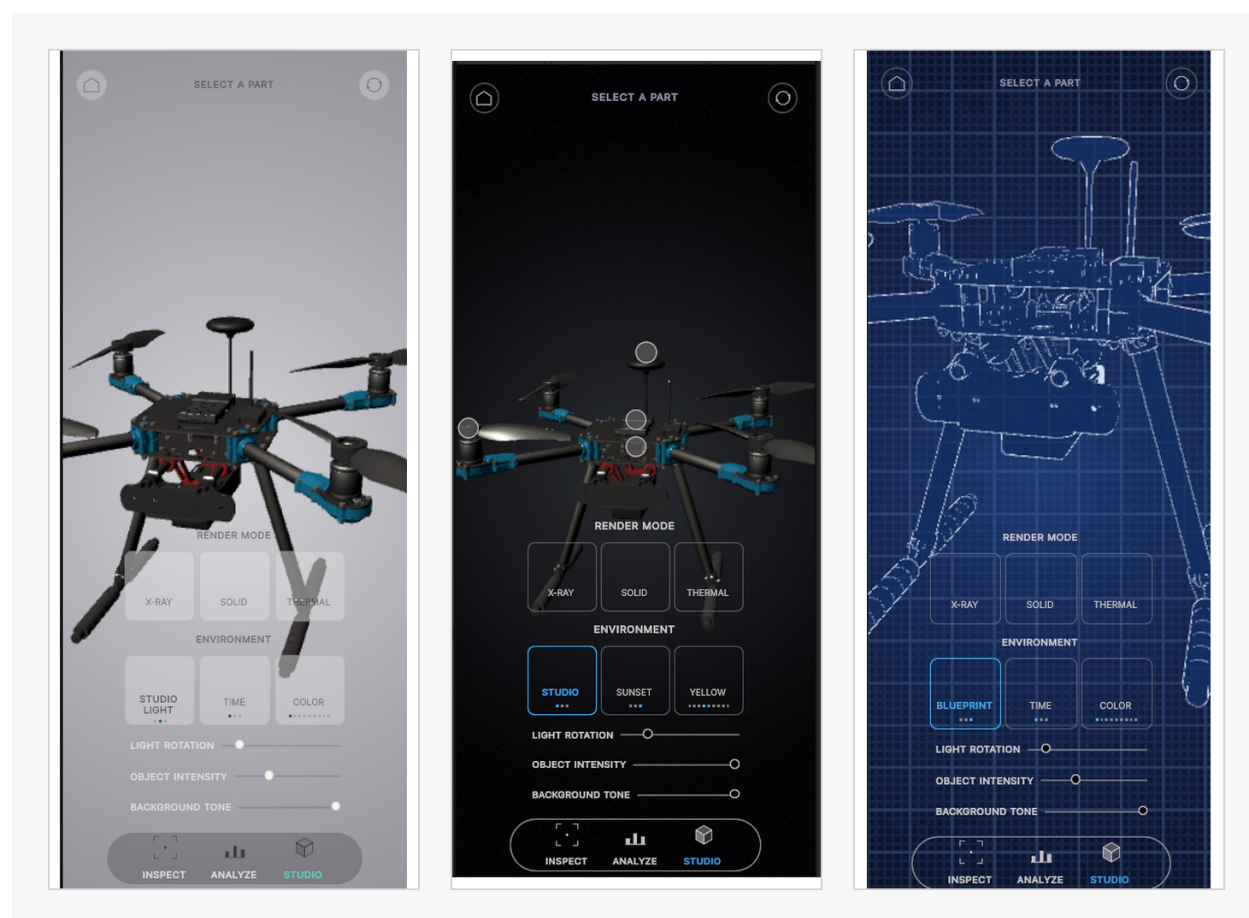
Modos directos de render para inspección visual del ensamblaje.



Nota. De izquierda a derecha: *X-Ray*, *Solid* y *Thermal*. Las capturas se agrupan porque corresponden al mismo panel de modos de render y mantienen proporción de pantalla móvil. Elaboración propia.

Figura 65

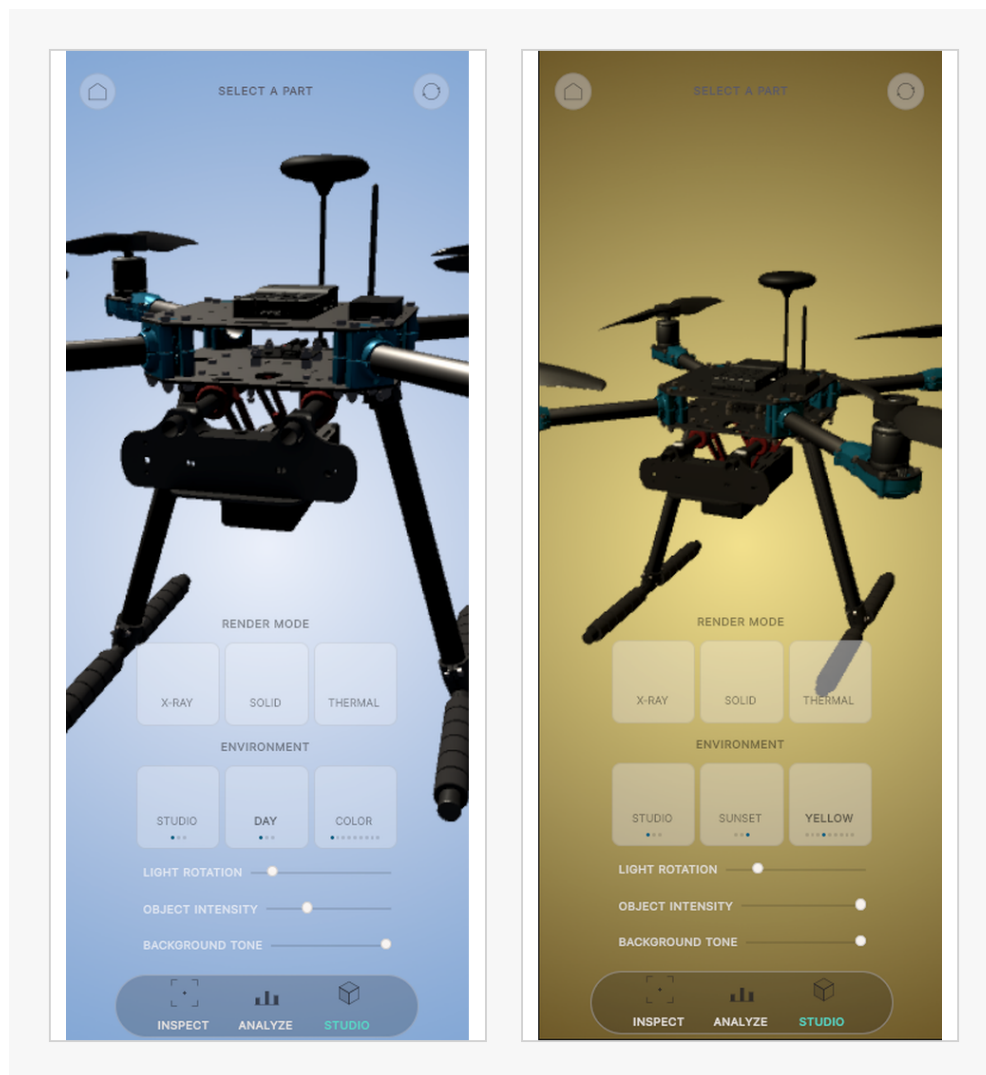
Presets de Studio y lectura Blueprint.



Nota. El preset *Blueprint* se activa desde *Studio* y muestra contornos, grilla y lectura técnica sin equivaler al modo oculto *Wireframe*. La lámina también documenta presets de presentación e iluminación del mismo panel. Elaboración propia.

Figura 66

Variación de entorno y color base desde Studio.



Nota. Las capturas ilustran cambios de ambiente y color base que afectan contraste, legibilidad del modelo y lectura de la interfaz sin modificar la geometría. Elaboración propia.

dron. Si la pieza no es fuente de calor, se toma:

$$T_{eq} = T_{amb}$$

Para fuentes térmicas, el equilibrio se interpola entre temperatura ambiente, temperatura de hover y temperatura pico según el factor de carga efectivo del sistema. La aproximación temporal hacia ese equilibrio se modela como:

$$\Delta T_{source} = (T_{eq} - T_{actual}) (1 - e^{-\Delta t/\tau}) w_s$$

donde τ es el tiempo de calentamiento configurado para la pieza y w_s es su peso relativo como fuente térmica.

El enfriamiento ambiental se aplica como:

$$\Delta T_{cooling} = (T_{amb} - T_{actual}) c_r \phi_{exp} \Delta t$$

donde c_r es la tasa de enfriamiento y ϕ_{exp} el factor de exposición convectiva.

La transferencia entre piezas conectadas por el grafo térmico se aproxima mediante:

$$\Delta T_{ij} = (T_j - T_i) \hat{G}_{ij} \Delta t$$

con un coeficiente de acoplamiento geométrico definido como:

$$\hat{G}_{ij} = \max \left(0,001, s_{ij} \frac{A_{ij}}{L_{ij}} \right)$$

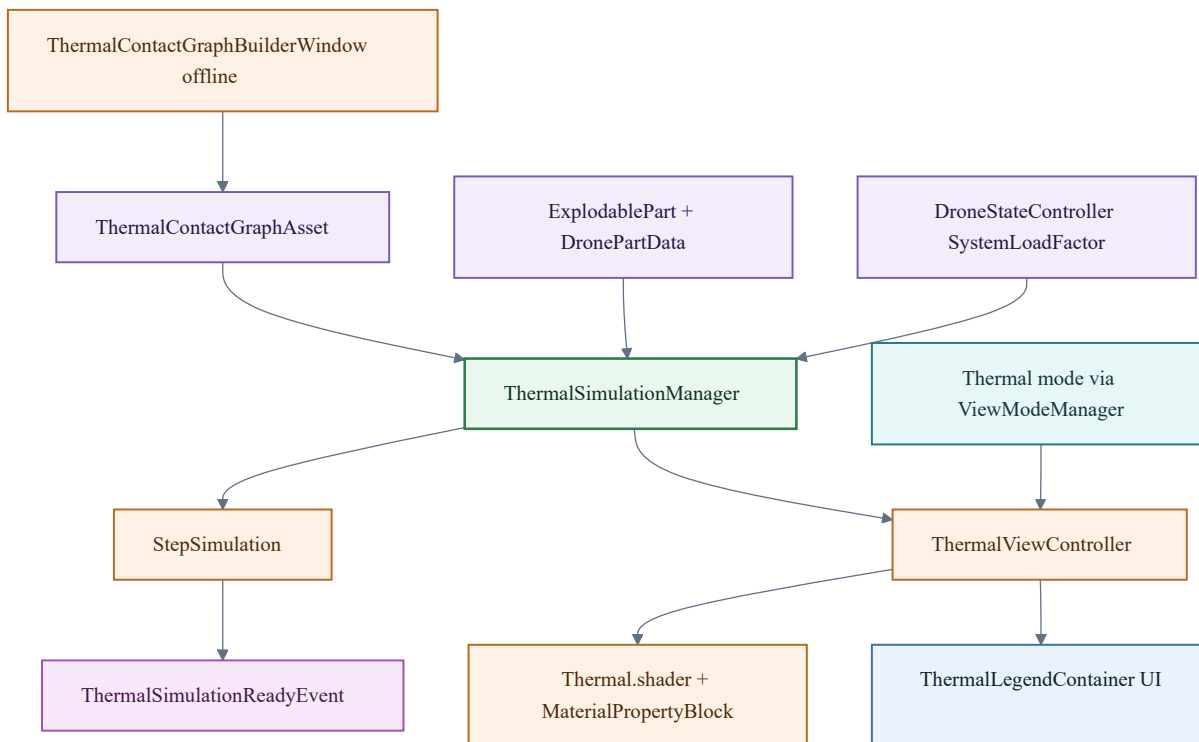
donde A_{ij} corresponde al área de contacto proxy, L_{ij} a la longitud efectiva del camino térmico y s_{ij} a un factor heurístico de enlace. En consecuencia, el sistema comparte la lógica de una conducción tipo Fourier, pero no debe interpretarse como una conductancia física calibrada en unidades SI.

La trazabilidad matemática del subsistema fue contrastada con el workflow de verificación basado en WolframAlpha utilizado por el proyecto. Dicho audit confirmó tres puntos: la estructura geométrica inspirada en A/L es razonable para el acoplamiento entre nodos, los tiempos de calentamiento del solver están deliberadamente comprimidos frente a valores físicos reales para mantener legibilidad interactiva, y la tasa de enfriamiento por defecto es coherente con esa misma escala temporal acelerada. Por ello, este subsistema debe documentarse como simulación térmica híbrida y heurística, no como FEA en tiempo real.

La Figura 67 articula la relación entre el estado del dron, la simulación térmica y la traducción visual del gradiente en la build final. A continuación, la Figura 68 muestra su traducción visual en la interfaz: un gradiente heurístico por componentes que orienta la inspección, sin representar termografía medida, FEA ni temperatura calibrada.

Figura 67

Arquitectura del subsistema térmico híbrido.



Nota. La figura ubica el modo *Thermal* como capa visual heurística por componentes, conectada al estado del dron y a un grafo térmico simplificado; no representa FEA, termografía ni temperatura real calibrada. Elaboración propia.

Para facilitar la comprensión del alcance y la organización del software interactivo desarrollado, los diferentes componentes del visor 3D se agrupan según su nivel de visibilidad y madurez en la build final (véase la Tabla 13).

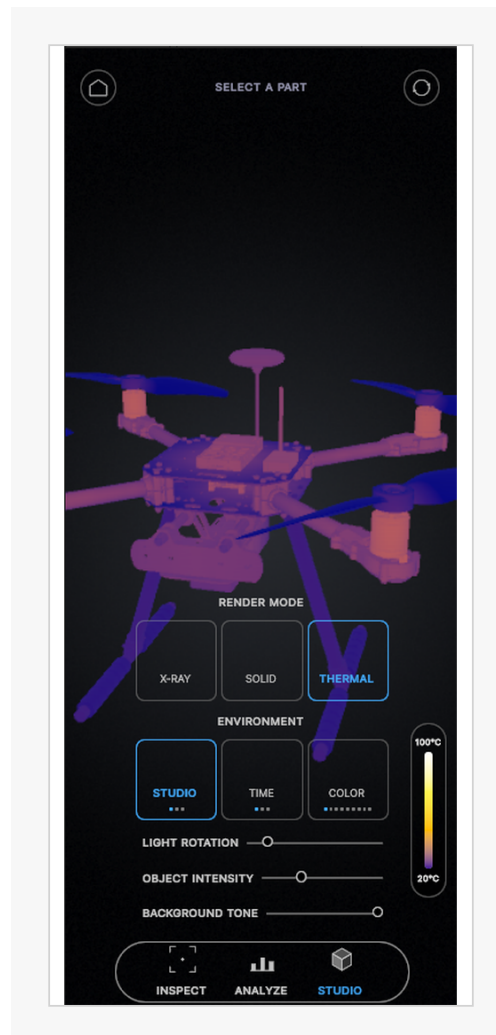
Matriz de estado funcional

Expuesto en UI final

- Hero y navegación principal;

Figura 68

Lectura visual del modo Thermal en la build final.



Nota. La captura muestra la convención visual usada para diferenciar componentes calientes y piezas pasivas dentro del modelo térmico heurístico. La leyenda orienta la lectura del gradiente sin representar termografía medida ni simulación FEA. Elaboración propia.

- selección de piezas y *bottom sheet*;
- hotspots;
- isolate;
- power y slider de carga;
- explode;
- cross-section;
- filtros por categoría;
- modos visuales *X-Ray*, *Solid View/Solid Color* y *Thermal*;
- presets de *Studio*, incluyendo *Blueprint*;
- thermal con leyenda visible.

Implementado pero oculto

- `MeasurementTool`;
- modos de vista *Wireframe* y *Ghosted* como capacidades implementadas pero no expuestas en la UI final;
- parte del flujo térmico avanzado y del clipping extendido.

Código legado o no integrado

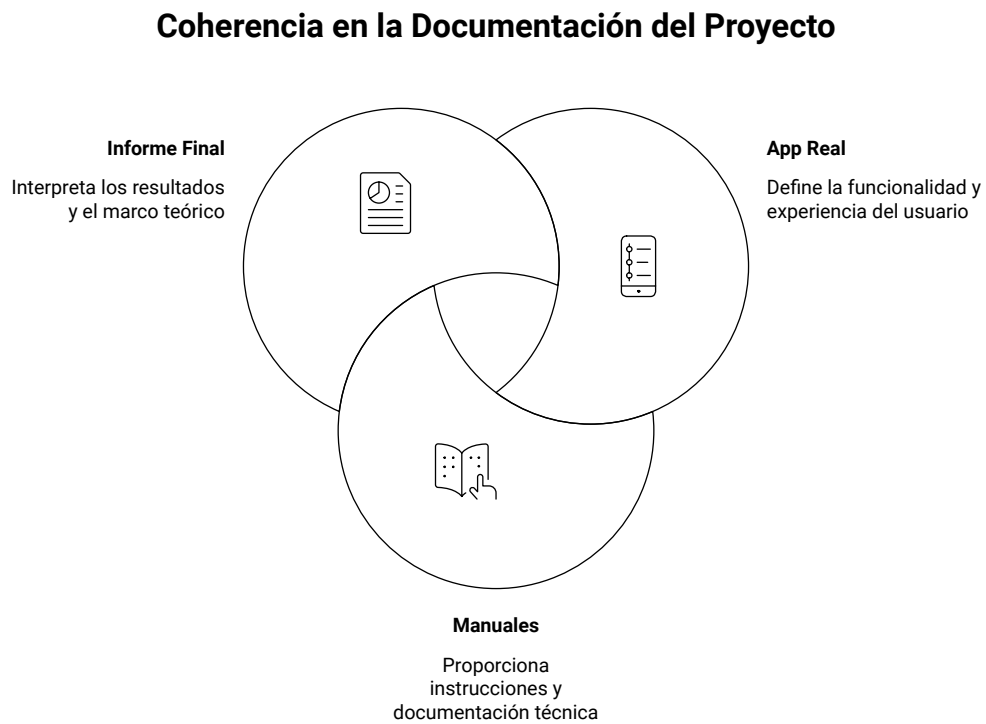
- `PartCatalogUI`;
- `SettingsPanel`;
- `LoadingController`;
- partes de `EnhancedInfoPanel`;
- suite de ensamblaje, BOM, anotaciones y connections.

Esta separación también asegura la coherencia documental entre la aplicación, los manua-

les y el presente informe (véase la Figura 69).

Figura 69

Coherencia documental entre app, manuales e informe.



Nota. El criterio de cierre separa funciones visibles, capacidades ocultas, código legado y trabajo futuro para evitar contradicciones entre la build y la documentación. Elaboración propia.

Tooling de editor, automatización y verificación

El proyecto conserva herramientas de editor y verificación que sí son defendibles como parte del desarrollo real:

- `ProjectSetupWizard;`
- `ImportedDroneCoverageAudit;`

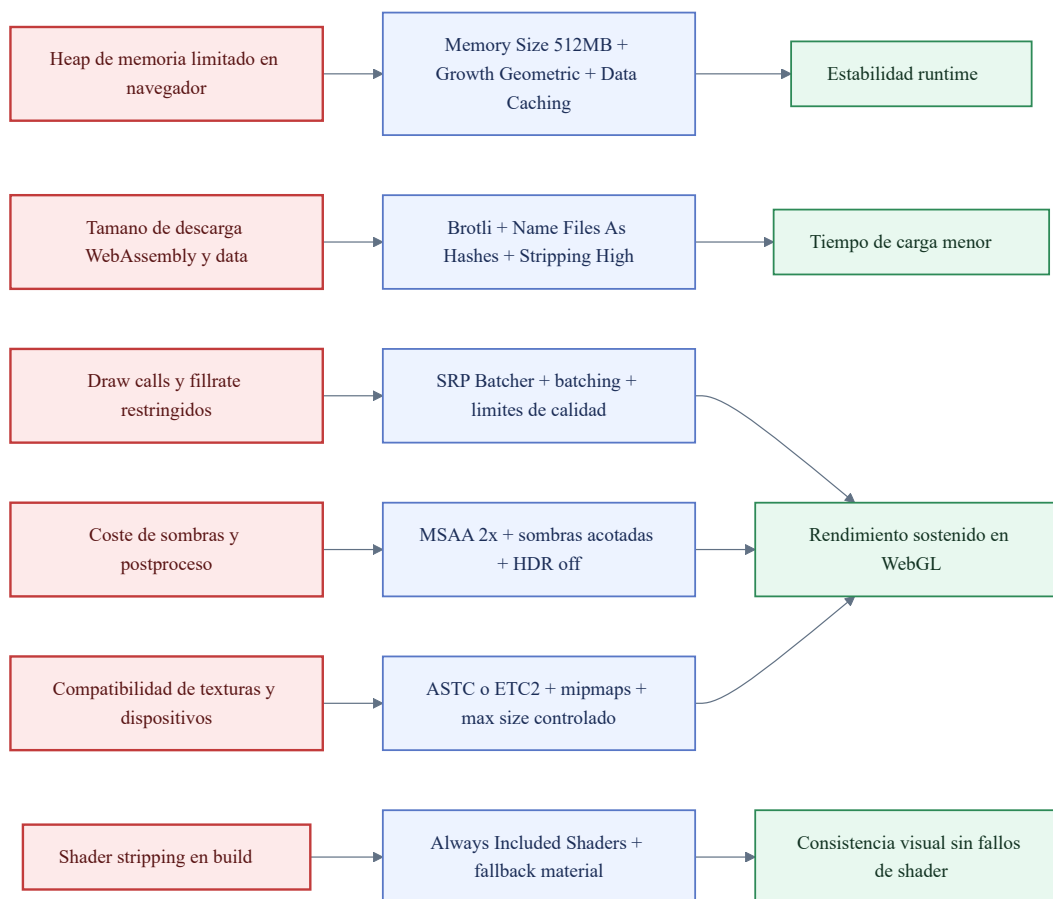
- `ThermalContactGraphBuilderWindow;`
- fixers de configuración y build WebGL.

Estas herramientas ayudan a mantener consistencia entre escena, catálogo, render y configuración de despliegue. No forman parte de la UI visible, pero sí del soporte técnico que hizo viable el proyecto.

Para lograr un despliegue viable en navegadores comerciales, el proyecto debió resolver las restricciones de memoria, procesamiento y acceso de la API WebGL. Las mitigaciones técnicas aplicadas incluyen el control estricto del presupuesto geométrico, la compactación de texturas en canales y la configuración óptima del tamaño de la memoria contigua de WebAssembly (véase la Figura 70).

Figura 70

Restricciones de WebGL y mitigaciones técnicas aplicadas al proyecto.

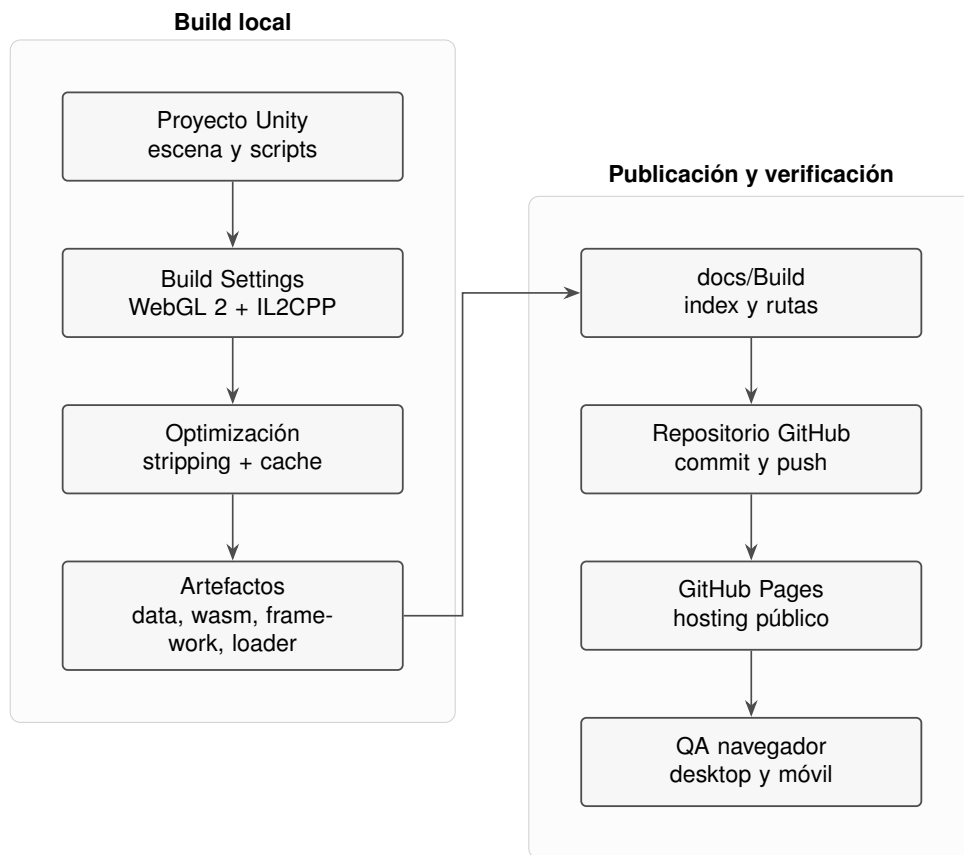


Nota. Resumen de los cuellos de botella característicos de la plataforma WebGL (memoria lineal, draw calls, descargas de red) y las correspondientes optimizaciones de Technical Art. Elaboración propia.

Una vez generada la build funcional de Unity Web, los archivos resultantes se integran en un pipeline de despliegue continuo alojado en GitHub Pages. Este flujo automatiza la compresión, publicación y verificación rápida del visor interactivo para la validación formativa y QA (véase la Figura 71).

Figura 71

Flujo de despliegue y artefactos desde la build hasta el hosting y QA.



Nota. Diagrama de flujo de integración continua, desde el empaquetado de assets hasta la publicación en el servidor de hosting y la evaluación de usuario. Elaboración propia.

Limitaciones y frentes de cierre

La build funcional y los ajustes P1/P2 quedan listos para entrega, evaluación y demostración. Los siguientes frentes delimitan el alcance documentado y separan el cierre verificable de las iteraciones futuras:

1. conservar como trazabilidad del pipeline la validación del FBX runtime final exporta-

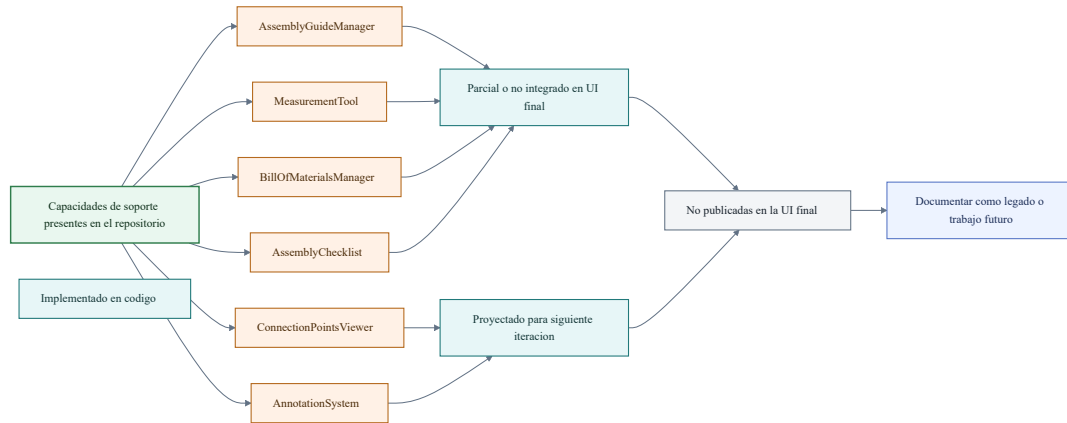
- do desde Blender, incluyendo masters, instancias, fasteners primitivos, hélices, hots-pots, filtros y grupos;
2. documentar materiales y texturas finales o, si no entran a la versión publicada, mantenerlos como mejora futura del pipeline visual;
 3. ejecutar rerun de auditorías jerárquicas y de cobertura si se reemplaza la escena o se hace un reimport mayor;
 4. decidir si el audio entra en una iteración posterior o queda como trabajo futuro;
 5. conservar la trazabilidad de métricas cuantitativas de rendimiento, SUS, NASA-TLX Raw y desempeño en tareas reportadas en el capítulo de resultados;
 6. decidir con datos de profiling si una ruta nativa de LOD se integra realmente al cierre del pipeline;
 7. reconocer que la interfaz de escritorio es una adaptación funcional del sistema *mobile-first*, no un sistema desktop rediseñado con el mismo nivel de rigor teórico;
 8. formalizar como trabajo futuro un *twin manifest* por pieza que conecte geometría, categoría, material, función, variables posibles, fallas, repuestos y procedimientos;
 9. mantener telemetría histórica, *live digital shadow*, modo servicio, diagnóstico y simulación calibrada como fases posteriores no implementadas en la build final.

Con estas delimitaciones, el sistema se documenta como funcional y verificable en su flujo principal, con evidencia técnica y de usuario reportada de forma descriptiva en el capítulo 5.

La Figura 72 deja explícito que el conjunto de herramientas de ensamblaje, BOM, anotaciones y conexiones pertenece a la capa proyectada o no integrada, por lo que no debe confundirse con el flujo visible de la build final.

Figura 72

Estado de herramientas de ensamblaje proyectadas o no integradas por módulo.



Nota. Módulos proyectados en la propuesta original que no fueron integrados en el flujo visible de la build final para acotar el alcance académico. Elaboración propia.

Resultados y Análisis

Este capítulo consolida los resultados funcionales, técnicos y de validación con usuarios obtenidos durante el cierre del proyecto. La evidencia se organiza en cuatro frentes: estado funcional de la build WebGL, rendimiento técnico por dispositivo, evaluación de usabilidad/carga de trabajo con 12 participantes anonimizados y resultados del pipeline de activos 3D. La lectura cuantitativa se mantiene descriptiva y formativa, coherente con el tamaño y tipo de muestra, pero se apoya en datos fechados y trazables de la build publicada.

Estado de cierre funcional

La build final publicada integra el flujo principal previsto para el MVP académico: ingreso desde el Hero, onboarding, exploración orbital, selección de piezas, *bottom sheet* de información, hotspots, aislamiento, modos *Inspect/Analyze/Studio*, corte transversal, vista explosionada, filtros por categoría y visualización térmica heurística. La revisión funcional posterior a las correcciones P1/P2 confirmó que el flujo visible corresponde con lo descrito en los manuales y con la arquitectura documentada en el capítulo de desarrollo.

La revisión funcional se complementa con mediciones técnicas y con instrumentos de usuario aplicados posteriormente. Por ello, la Tabla 14 resume el cierre operativo de la app, mientras que las subsecciones siguientes reportan los datos de rendimiento, SUS, NASA-TLX Raw y *Think-Aloud*.

Aporte frente a necesidades de la industria de hardware complejo

La revisión conceptual de cierre permitió reinterpretar los resultados funcionales del prototipo frente a necesidades recurrentes de la industria de hardware complejo. El producto no debe evaluarse únicamente como una escena 3D, sino como una primera capa de comprensión espa-

Tabla 13*Inventario sintético de módulos y estado de integración*

Categoría	Componentes representativos	Estado	Interpretación
Runtime principal	Orquestación UI, selección, explode, clipping, view modes, preset <i>Blueprint</i> y térmico	Expuesto o activo en la build final	Se documenta en el cuerpo principal porque soporta el flujo visible y la arquitectura operativa.
Capacidades ocultas	MeasurementTool; modos <i>Wireframe</i> y <i>Ghosted</i> ; clipping extendido; térmico avanzado	Implementado pero oculto	Puede mencionarse como capacidad existente, pero no debe narrarse como funcionalidad publicada al usuario final.
Legado o no integrado	Catálogo legado, SettingsPanel, LoadingController, paneles legacy y suite de ensamblaje	No integrado en el flujo final	Debe permanecer separado como legado, trabajo futuro o evidencia histórica de evolución del proyecto.
Tooling de editor	Wizard de proyecto, auditoría de cobertura y constructor del grafo térmico	Soporte técnico de desarrollo	Es pertinente para la defensa técnica y la trazabilidad del pipeline, aunque no forme parte de la UI publicada.

Nota. Esta tabla resume el criterio usado para separar funciones visibles, funciones ocultas y módulos no integrados, evitando mezclar capacidades de desarrollo con el alcance real de la build final. Elaboración propia a partir del código y de la validación funcional del proyecto.

Tabla 14*Estado de cierre funcional por frente de validación*

Frente	Estado	Evidencia de cierre	Dato de validación
Flujo principal WebGL	QA funcional correcto	La app permite explorar, seleccionar, aislar, analizar y volver al menú sin romper el flujo visible	Métricas por escenario
Interacción móvil	QA P1/P2 correcto	Gestos, <i>bottom sheet</i> , doble toque, zoom, órbita e iconos fueron ajustados tras observación directa	Tabla por dispositivo
Rendimiento percibido	Correcto en revisión formativa	La app se observó fluida en escritorio y en la mayoría de móviles de gama media-baja probados, incluyendo Redmi Note 10S	FPS, memoria y carga
Usabilidad formativa	Correcta para cierre iterativo	La prueba breve permitió convertir gestos espontáneos en ajustes concretos de interfaz	SUS, NASA-TLX y <i>Think-Aloud</i>
Manuales e informe	Coherencia revisada	La documentación distingue funciones visibles, capacidades ocultas y resultados de validación	Datos de campo registrados

Nota. Esta tabla resume el cierre funcional y no reemplaza las tablas cuantitativas posteriores. Elaboración propia.

cial, comunicación técnica y preparación semántica de componentes. En ese sentido, la build final cubre parte de las necesidades de inspección, soporte inicial, entrenamiento y demostración del producto, aunque todavía no cubre telemetría, diagnóstico, mantenimiento guiado ni interoperabilidad industrial completa (véanse la Tabla 15 y la Figura 73).

Tabla 15

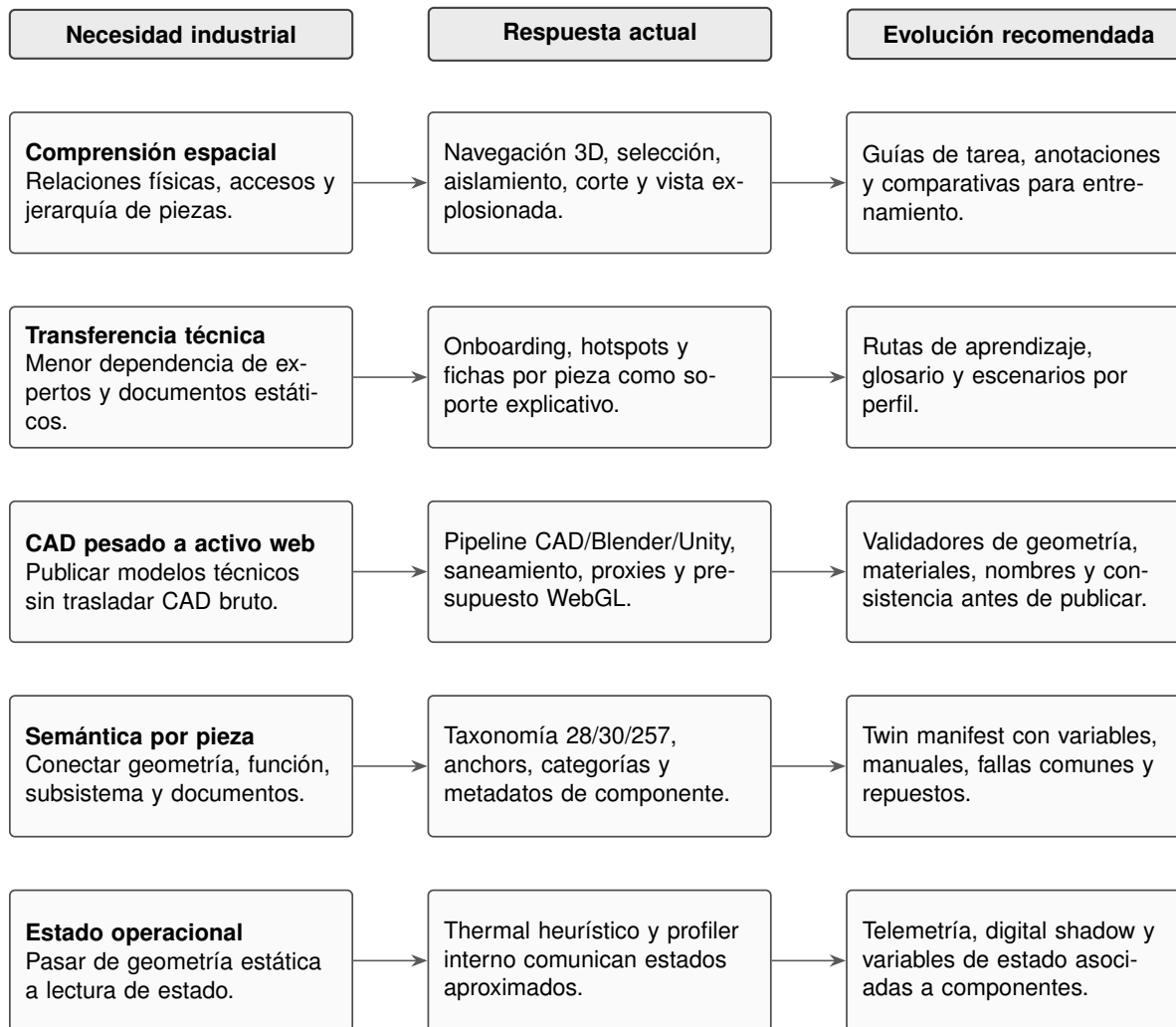
Relación entre necesidades industriales y respuesta del prototipo

Necesidad industrial	Respuesta actual del proyecto	Límite vigente	Madurez
Comprensión espacial de hardware complejo	Navegación 3D, selección, aislamiento, vista explosionada, corte y modos de inspección	Se complementa con documentación técnica, validación por tareas e instrumentos de usuario	Visual product twin
Transferencia de conocimiento técnico	Onboarding, hotspots, fichas por pieza, categorías y flujo mobile-first	Falta medir formalmente aprendizaje longitudinal o retención posterior	Visual product twin
Conversión CAD a activo web	Pipeline CAD/Blender/Unity, saneamiento runtime, presupuesto geométrico y profiler interno	La reducción geométrica se reporta por rutas de importación y optimización	Digital model semántico
Datos de producto fragmentados	Taxonomía 28/30/257, grupos de escena y fichas como base de identificadores	No existe aún <i>twin manifest</i> formal ni conectores externos	Preparación semántica
Soporte y mantenimiento	Selección contextual y lectura por componentes pueden servir como base de procedimientos	No hay síntomas, repuestos, historial, órdenes de trabajo ni reglas de diagnóstico	Trabajo futuro
Estado operacional y telemetría	Visualización térmica heurística introduce la idea de estado por componente	No hay telemetría real, calibración física ni sincronización con activo	Pre-shadow conceptual

Nota. La tabla distingue respuesta funcional actual, límite técnico y nivel de madurez. No clasifica la build como *digital twin* operacional. Elaboración propia a partir de la investigación conceptual y la build final.

Figura 73

Necesidades industriales, respuesta actual y evolución futura defendible.



Nota. La lectura va de izquierda a derecha: necesidad industrial, capacidad ya entregada y evolución recomendada.

La figura no clasifica la build actual como *digital twin* operacional. Elaboración propia.

Esta lectura fortalece la justificación del proyecto: el dron Holybro X500 V2 opera como caso representativo de un sistema electromecánico multicomponente, pero el problema de fondo es transferible a maquinaria, robótica, equipos técnicos, plataformas motorizadas e instrumentos

con integración mecánica, eléctrica, electrónica, sensorial y de control. La contribución actual consiste en demostrar una capa visual-semántica usable y desplegable en web; la contribución futura dependerá de conectar esa capa con datos operativos y procedimientos de servicio.

Resultados formativos de rendimiento

El cierre técnico incorporó un profiler interno capaz de registrar sesiones por escenario y exportarlas en JSON o CSV. Cada sesión permite asociar la experiencia de uso con FPS promedio, *frame time*, peor *frame time*, memoria, resolución, plataforma, dispositivo, navegador, modo visual, factor de explosión, corte transversal, cantidad de renderers, mallas y triángulos estimados. Esta instrumentación resuelve el problema metodológico de depender únicamente de impresiones subjetivas o de herramientas externas.

Durante la revisión directa de la build se observó que la optimización es adecuada para el alcance del MVP: la aplicación se mantuvo fluida en escritorio y funcional en móviles reales de gama media, con Redmi Note 10S como referencia representativa compartida durante el cierre. Esta afirmación se limita al comportamiento observado y se complementa con el resumen por dispositivo de la Tabla 16, con los KPIs complementarios de la Tabla 17 y con el detalle de sesiones de la Tabla 18. Estas tablas proceden de exportaciones fechadas del profiler interno de la app y de artefactos publicados de la build, por lo que separan medición técnica, lectura interpretativa y exclusiones justificadas. En los casos en que Unity WebGL normaliza el sistema operativo en la exportación del profiler, la identificación del entorno se contrasta con el registro manual de dispositivo, sistema operativo, navegador y resolución.

Tabla 16*Resumen de rendimiento WebGL por dispositivo y escenario operativo*

Dispositivo	SO / navegador	Escenarios válidos	Dur. (s)	FPS prom.	Frame prom.	Pico mem.	Lectura
PC escritorio (Intel Core i7-5820K, GTX 980 Ti, 48 GB RAM), 1910×903	Windows 11 / Chrome 141	Base, selección/aislamiento, explode, cut, thermal/studio	253.4	59.8	16.7 ms	413 MB	Fluidez estable en todos los escenarios operativos.
Móvil Android (Adreno 610), 720x1300	Android 10 / Chrome 148	Base, selección/aislamiento, explode, cut, thermal/studio	389.5	24.5	52.1 ms	311 MB	Rendimiento usable, con picos perceptibles y thermal/studio como escenario más exigente.
Móvil Android (Adreno 610), 720x1410	Android 12 / Chrome 114	Base, selección/aislamiento, explode, cut, thermal/studio	115.4	17.6	61.4 ms	263 MB	Límite inferior móvil: navegable, pero por debajo de la meta de 30 FPS.
Móvil Android (Adreno 650), 1280x1914	Android 12 / Chrome 114	Base, selección/aislamiento, explode, cut, thermal/studio	149.8	30.0	36.2 ms	254 MB	Cerca de la meta de 30 FPS, con picos aislados durante cut.
Redmi Note 10S (Mali-G76), 924x1862	MIUI Global 14.0.11 / Chrome 148	Base, selección/aislamiento, thermal/studio, explode, cut	156.7	26.5	40.3 ms	226 MB	Experiencia móvil funcional en el flujo principal.
iPhone 17 Pro (A19 Pro), 1206x2622	iOS / Safari	Base, selección/aislamiento, explode, cut, thermal/studio	180.0	58.7	17.0 ms	356 MB	Fluidez cercana al objetivo de 60 FPS en recorrido completo.
Móvil Android (Adreno 650), 1280x1808	Android 12 / Chrome 91	Base	–	–	–	119 MB	Captura descartada del agregado por comportamiento atípico de pausa o suspensión.

Nota. Los promedios se calcularon sobre sesiones operativas exportadas por el profiler interno de la app. Se excluyeron segmentos de arranque corto, capturas accidentales y una captura atípica de Chrome 91 con peor frame de 87699 ms. Elaboración propia a partir de las mediciones técnicas WebGL del 4 de junio de 2026.

Tabla 17*KPIs complementarios de build, memoria y escena runtime*

Indicador	Valor reportado	Lectura para validación
Payload activo publicado	15.78 MB (15.05 MiB)	Suma de los artefactos activos referenciados por docs/Build/index.html: .data, .wasm, framework y loader.
Archivo .data.unityweb	8.48 MB	Huella principal de datos iniciales del visor WebGL publicado.
Archivo .wasm.unityweb	7.11 MB	Código WebAssembly comprimido asociado a la build Unity Web.
Framework + loader	0.19 MB	Capa JavaScript de arranque y puente de carga del runtime web: 0.08 MB de framework y 0.12 MB de loader.
Heap WebGL configurado	32 MB inicial / 512 MB máximo	Configuración de Unity WebGL con crecimiento geométrico habilitado para escalar memoria sin reservar el máximo desde el inicio.
Crecimiento de heap	Paso geométrico 20 %, cap 96 MB	Estrategia de crecimiento registrada en ProjectSettings.asset; reduce riesgo de arranque pesado y permite expansión controlada.
Pico de memoria reservada observado	413 MB escritorio; 311 MB Android Adreno 610; 226 MB Redmi Note 10S; 356 MB iOS	El pico se interpreta como indicador comparativo por sesión y no como memoria física total del dispositivo.
Escena runtime instrumentada	252 renderers, 252 mallas, 229 054 triángulos estimados	Conteo exportado por el profiler interno para asociar rendimiento observado con complejidad renderizable de la escena evaluada.
Primera captura operativa del profiler	2.8 s escritorio; 3.8–4.1 s Android; 27.0 s Redmi Note 10S	Tiempo hasta inicio de sesión de profiler interno, usado como referencia operativa de captura y no como cronometraje de tareas de usuario.

Nota. Los artefactos corresponden a la build publicada en docs/Build y fueron recalculados sobre los archivos activos generados el 10 de junio de 2026: 6cd0d03c140b38df116217ef412de022.data.unityweb, 6b3071039ac238a7bac2d1c430852e0e.wasm.unityweb, 00424fe2ea5640673e2c8862c0cca28e.framework.js.unityweb y c02a0bc936202d8ae347ae331e242c98.loader.js. Los parámetros de heap provienen de la configuración WebGL del proyecto Unity y las cifras de escena runtime provienen de las exportaciones JSON/CSV del profiler interno. Elaboración propia.

Tabla 18*Detalle de sesiones de rendimiento WebGL por dispositivo y escenario*

Dispositivo	Escenario	Dur. (s)	FPS	Frame ms	Peor ms	Pico mem.	Uso
PC escritorio (GTX 980 Ti)	Base	5.8	54.5	24.6	93.7	119 MB	Excluida: arranque corto
PC escritorio (GTX 980 Ti)	Base	24.7	59.9	16.7	17.8	146 MB	Agregada
PC escritorio (GTX 980 Ti)	Selección/aislamiento	31.4	59.0	17.0	20.9	342 MB	Agregada
PC escritorio (GTX 980 Ti)	Explode	38.5	59.8	16.7	18.5	371 MB	Agregada
PC escritorio (GTX 980 Ti)	Cut / corte	65.9	60.0	16.7	17.2	372 MB	Agregada
PC escritorio (GTX 980 Ti)	Thermal/Studio	92.9	59.8	16.7	18.6	413 MB	Agregada
Móvil Android (Adreno 610)	Base	93.7	25.0	73.3	1468.0	120 MB	Agregada

Dispositivo	Escenario	Dur. (s)	FPS	Frame ms	Peor ms	Pico mem.	Uso
Móvil Android (Adreno 610)	Base	86.4	25.9	40.3	67.9	205 MB	Agregada
Móvil Android (Adreno 610)	Selección/aislamiento	71.3	24.6	46.3	218.5	280 MB	Agregada
Móvil Android (Adreno 610)	Explode	29.5	26.3	38.4	50.1	309 MB	Agregada
Móvil Android (Adreno 610)	Cut / corte	40.5	26.6	38.5	62.5	309 MB	Agregada
Móvil Android (Adreno 610)	Thermal/Studio	68.1	20.0	57.9	303.0	311 MB	Agregada
Móvil Android (Adreno 610, Chrome 114)	Base	7.9	17.6	172.9	911.8	118 MB	Excluida: arranque corto
Móvil Android (Adreno 610, Chrome 114)	Base	9.9	15.7	71.6	139.5	157 MB	Agregada
Móvil Android (Adreno 610, Chrome 114)	Selección/aislamiento	25.2	20.0	54.3	147.0	233 MB	Agregada
Móvil Android (Adreno 610, Chrome 114)	Explode	22.0	18.8	53.9	70.9	261 MB	Agregada
Móvil Android (Adreno 610, Chrome 114)	Cut / corte	20.3	18.7	56.3	126.2	262 MB	Agregada
Móvil Android (Adreno 610, Chrome 114)	Thermal/Studio	38.0	15.3	70.5	136.3	263 MB	Agregada
Móvil Android (Adreno 650)	Base	4.8	32.3	73.9	264.6	118 MB	Excluida: arranque corto
Móvil Android (Adreno 650)	Base	20.4	32.8	31.7	40.8	149 MB	Agregada
Móvil Android (Adreno 650)	Selección/aislamiento	18.0	37.2	28.1	47.4	199 MB	Agregada
Móvil Android (Adreno 650)	Explode	19.4	29.3	34.4	43.1	225 MB	Agregada
Móvil Android (Adreno 650)	Cut / corte	41.3	32.6	33.8	142.7	226 MB	Agregada
Móvil Android (Adreno 650)	Thermal/Studio	50.7	24.4	43.4	58.8	254 MB	Agregada
Redmi Note 10S (Mali-G76)	Base	17.7	29.0	38.3	126.0	157 MB	Agregada
Redmi Note 10S (Mali-G76)	Base	15.2	30.7	33.4	43.6	157 MB	Agregada
Redmi Note 10S (Mali-G76)	Selección/aislamiento	26.3	30.9	34.3	104.0	221 MB	Agregada
Redmi Note 10S (Mali-G76)	Thermal/Studio	38.7	21.5	48.0	88.8	225 MB	Agregada
Redmi Note 10S (Mali-G76)	Base	0.9	21.3	47.1	48.5	225 MB	Excluida: segmento corto
Redmi Note 10S (Mali-G76)	Explode	28.1	26.5	38.6	60.2	225 MB	Agregada
Redmi Note 10S (Mali-G76)	Cut / corte	30.8	25.6	41.8	100.6	226 MB	Agregada
iPhone 17 Pro (A19 Pro, Safari)	Recorrido completo	180.0	58.7	17.0	24.8	356 MB	Agregada
Móvil Android (Adreno 650, Chrome 91)	Base	406.4	21.6	2369.6	87699.0	119 MB	Excluida: captura atípica

Nota. El campo “Pico mem.” corresponde al pico de memoria reservada reportado por el profiler interno. La memoria del navegador WebGL debe interpretarse como indicador comparativo de la sesión, no como memoria física total del dispositivo. Elaboración propia.

Las métricas runtime deben leerse junto con el dispositivo, el navegador, la resolución,

el estado de caché y el escenario activo. La cifra de 95 617 triángulos corresponde al activo base optimizado exportado; la cifra de 229 054 triángulos estimados corresponde al conteo runtime reportado por el profiler para la escena instrumentada. No son métricas equivalentes y no deben compararse de forma directa, porque la ejecución instrumentada incorpora instancias, proxies, assets de apoyo y renderers adicionales.

En conjunto, las mediciones móviles son prometedoras si se consideran las restricciones de hardware de equipos de gama media-baja. El prototipo no alcanza 30 FPS en todos los escenarios de los dispositivos Android de límite inferior, pero se aproxima a esa meta en flujos principales y la supera o se mantiene cerca en equipos de gama media-alta y alta. Esta lectura indica que la reducción geométrica, el control de materiales y la preparación WebGL cumplen el comportamiento esperado para un MVP interactivo de hardware complejo, aunque no convierten la compatibilidad móvil en una garantía universal para cualquier dispositivo.

Adicionalmente, la compatibilidad del prototipo se evaluó considerando diferentes factores de forma y navegadores (véanse la Tabla 19 y la Figura 74).

Tabla 19*Compatibilidad por factor de forma, navegador y estado de validación*

Factor de forma	Navegador	Compatibilidad esperada	Validado	Observaciones
Escritorio	Windows 11 / Chrome 141	Compatible	Sí	Fluidez estable en escenarios base, selección, explode, corte y thermal/studio.
Móvil Android gama media-baja	MIUI Global 14.0.11 / Chrome 148	Compatible con restricciones WebGL	Sí	Usable con picos perceptibles; referencia adicional para Redmi Note 10S.
Móvil Android límite inferior	Android 12 / Chrome 114	Compatible con restricciones	Sí	Navegable, pero por debajo de la meta de 30 FPS en la sesión registrada.
Móvil Android gama media-alta	Android 12 / Chrome 114	Compatible	Sí	Cercano a 30 FPS promedio y estable para el flujo principal.
Móvil iOS gama alta	iOS / Safari	Compatible	Sí	Fluidez cercana a 60 FPS en recorrido completo.
Captura descartada	Android 12 / Chrome 91	No usada para agregado	No	Sesión separada por pausa o suspensión atípica del navegador.

Nota. La tabla resume compatibilidad observada por factor de forma y navegador a partir de las sesiones del profiler interno. Elaboración propia.

Figura 74

Matriz visual de compatibilidad por factor de forma y dispositivo.

Dispositivo	SO / navegador	Resolución	FPS prom.	Lectura
PC escritorio	Windows 10 / Chrome 141	1910x903	59.8 FPS	Estable
Android Adreno 610	Android 10 / Chrome 148	720x1300	24.5 FPS	Usable
Android Adreno 610	Android 12 / Chrome 114	720x1410	17.6 FPS	Límite inferior
Android Adreno 650	Android 12 / Chrome 114	1280x1914	30.0 FPS	Cerca de meta
Redmi Note 10S	Android 10 / Chrome 148	924x1862	26.5 FPS	Funcional
iPhone 17 Pro	iOS / Safari	1206x2622	58.7 FPS	Cercano a 60 FPS
Android Chrome 91	Android 12 / Chrome 91	1280x1808	Excluida	Pausa atípica

Nota. La matriz sintetiza dispositivo, sistema operativo, navegador, resolución, FPS promedio y lectura operativa de las sesiones técnicas registradas. Elaboración propia.

Ajustes formativos previos a la evaluación con usuarios

Antes de aplicar los instrumentos con la muestra final se realizó una prueba breve de *Think-Aloud* y observación de acciones espontáneas. El foco no fue obtener un puntaje estadístico, sino detectar gestos que los usuarios intentaban ejecutar de forma automática. Esta decisión fue clave porque varias correcciones finales no surgieron de fallos técnicos evidentes, sino de expectativas naturales de interacción:

- deslizar tarjetas del tutorial;
- arrastrar verticalmente el panel inferior;
- controlar el zoom con menor sensibilidad;
- aislar piezas sin que apareciera de inmediato un panel de información;
- usar doble toque sobre piezas u hotspots.

La revisión permitió convertir esas observaciones en ajustes concretos: el tutorial acepta gestos de deslizamiento y cierre, el panel de información bloquea su área táctil para evitar interacción accidental con el dron, el zoom y la órbita redujeron sensibilidad, el doble toque de aislamiento se estabilizó para piezas y hotspots, la navegación con Atrás se organizó por capas y la salida de la app se reserva para el menú principal con confirmación. Estos cambios fortalecieron la versión evaluada con los instrumentos SUS, NASA-TLX Raw y *Think-Aloud* (véanse la Tabla 20 y la Figura 75).

Tabla 20

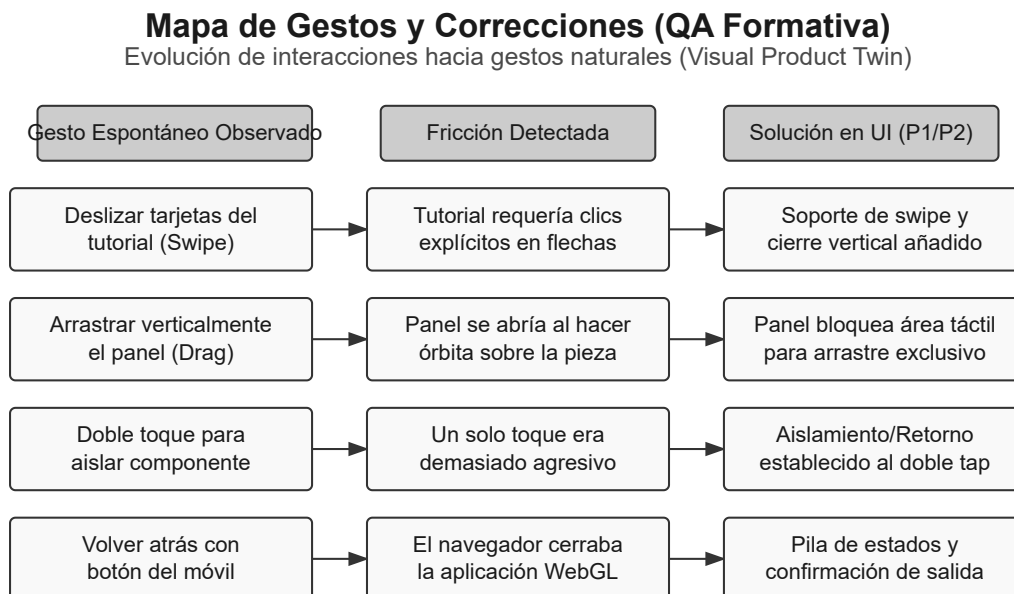
Hallazgos formativos y correcciones aplicadas antes de la evaluación con usuarios

Observación	Comportamiento detectado	Corrección aplicada	Estado
Tutorial	Usuarios intentaron hacer <i>swipe</i> entre tarjetas y cerrar con gesto descendente	Se añadió navegación por dots/gestos y cierre del tutorial con deslizamiento hacia abajo	QA correcto
Panel de información	El gesto de abrir/cerrar se confundía con interacción sobre el dron	El <i>bottom sheet</i> bloquea su rectángulo táctil y prioriza el gesto vertical del panel	QA correcto
Zoom y órbita	La sensibilidad táctil se percibió demasiado alta	Se redujo la sensibilidad de zoom y órbita en móvil	QA correcto
Aislamiento	Abrir el panel automáticamente al aislar resultó agresivo	Doble toque aísla/desaísla sin abrir panel de información	QA correcto
Hotspots	En móvil se esperaba doble toque también sobre hotspots	Se estabilizó doble tap para aislamiento de hotspots	QA correcto
Botón Atrás	El navegador tendía a salir de la app sin respetar estado interno	Se implementó navegación por capas y confirmación de salida desde menú principal	QA correcto
Iconos móviles	Algunas microinteracciones quedaban en estados visuales incorrectos	Se auditaron y estabilizaron estados táctiles de iconos procedurales	QA correcto

Nota. Estos hallazgos provienen de observación formativa previa y explican la evolución P1/P2 aplicada antes de consolidar los instrumentos de usuario. Elaboración propia.

Figura 75

Mapa de gestos espontáneos y correcciones P1/P2.



Nota. Mapeo visual que conecta las fricciones de interacción táctil identificadas en la prueba previa de Think-Aloud con sus correspondientes soluciones implementadas en la interfaz de la build. Elaboración propia.

Resultados de usabilidad

La evaluación de usabilidad se aplicó a 12 participantes anonimizados entre el 20 de mayo y el 3 de junio de 2026. Diez participantes realizaron la prueba en smartphone y dos participantes la realizaron en PC; en todos los casos se utilizó Google Chrome. Las pruebas móviles se ejecutaron en un Redmi Note 10S con MIUI Global 14.0.11 y Google Chrome 148. Las dos sesiones de escritorio se realizaron en el mismo equipo empleado para las pruebas técnicas: Intel Core i7-5820K, NVIDIA GTX 980 Ti, 48 GB RAM, resolución efectiva del viewport de 1910×903, Windows 11 y Google Chrome 141. La evaluación con usuarios se ejecutó so-

bre la misma build usada para la recolección de datos técnicos; en la copia pública de cierre, dicha build queda trazada por la rama *master*, la URL publicada y los artefactos versionados en *docs/Build*, *Informe_final/* y *Telemetria/Mediciones_WebGL*. En todas las sesiones se utilizó la caché del navegador previamente cargada. La secuencia de exposición se contrabalanceó entre visor 3D primero (AB) y soporte 2D primero (BA). La matriz de desempeño registró cuatro tareas equivalentes por condición: ubicación de componente, interpretación funcional, relación espacial y análisis visual guiado. En total se documentaron 96 registros tarea-condición; todos fueron completados, con una ayuda baja en la condición 3D durante la tarea de interpretación funcional del participante 006 (véase la Tabla 21).

Tabla 21

Desempeño observado por tarea y condición

Tarea	Visor 3D	Soporte 2D	Errores / ayudas	Lectura
T1. Ubicación de componente	12/12 completadas	12/12 completadas	0 errores; 0 ayudas	Los participantes localizaron el motor y piezas cercanas en ambas condiciones.
T2. Interpretación funcional	12/12 completadas	12/12 completadas	0 errores; 1 ayuda baja en 3D	El <i>bottom sheet</i> permitió consultar información de pieza; un caso requirió repetir la instrucción.
T3. Relación espacial	12/12 completadas	12/12 completadas	0 errores; 0 ayudas	La relación motor–montura–tornillería fue resuelta en ambos medios, con mayor claridad cualitativa atribuida al visor 3D.
T4. Análisis visual guiado	12/12 completadas	12/12 completadas	0 errores; 0 ayudas	En 3D se usaron vista explosionada, <i>X-Ray</i> y presets visuales; en 2D se interpretó la información estática equivalente.

Nota. La tabla sintetiza los registros estructurados del moderador por tarea y condición. La lectura cualitativa complementaria se presenta en la sección de *Think-Aloud*. Elaboración propia.

La medición temporal se aplicó a T1, T2 y T3, porque estas tareas tenían inicio y cierre claramente comparables entre visor 3D y soporte 2D. La T4 se mantuvo como tarea exploratoria guiada y, por esa razón, no se cronometró ni se incluye en el análisis temporal. Su lectura se conserva en la observación cualitativa y en el protocolo *Think-Aloud*. Los tiempos se registraron co-

mo cronometraje operativo en segundos durante la sesión moderada; por ello se interpretan como evidencia descriptiva de desempeño y no como medición instrumental de alta precisión temporal.

Tabla 22

Tiempos registrados por participante en tareas cronometradas

Código	Sec.	Disp.	T1 3D	T1 2D	T2 3D	T2 2D	T3 3D	T3 2D
P001	AB	Smartphone	5	13	3	18	15	23
P002	BA	Smartphone	5	13	3	18	11	23
P003	AB	Smartphone	5	13	3	18	11	23
P004	BA	PC	5	13	3	18	11	23
P005	AB	Smartphone	5	13	9	18	11	23
P006	BA	Smartphone	5	13	3	18	11	23
P007	AB	Smartphone	5	13	3	18	11	23
P008	BA	Smartphone	5	13	3	18	11	23
P009	AB	Smartphone	14	13	3	18	11	23
P010	BA	Smartphone	5	13	3	18	11	23
P011	AB	Smartphone	5	13	3	18	11	23
P012	BA	PC	5	13	3	18	11	23

Nota. Los tiempos se presentan en segundos. Sec. = secuencia de exposición; AB = visor 3D primero y soporte 2D después; BA = soporte 2D primero y visor 3D después. Elaboración propia a partir de los registros del moderador.

Tabla 23

Resumen descriptivo de tiempos en tareas cronometradas

Tarea	Media 3D	Media 2D	Dif. 2D–3D	n	Lectura
T1	5,75 s	13,00 s	7,25 s	12	Favorable al visor 3D.
T2	3,50 s	18,00 s	14,50 s	12	Favorable al visor 3D.
T3	11,33 s	23,00 s	11,67 s	12	Favorable al visor 3D.
Total T1–T3	20,58 s	54,00 s	33,42 s	12	Menor duración acumulada en 3D.

Nota. La diferencia se calcula como media del soporte 2D menos media del visor 3D. Elaboración propia.

En las tres tareas cronometradas, la condición 3D presentó menor tiempo medio que el soporte 2D. El tiempo total medio acumulado en T1–T3 fue de 20,58 s para el visor 3D y de 54,00

s para el soporte 2D, con una diferencia media de 33,42 s a favor del visor. Estos resultados se interpretan de forma descriptiva y complementan la lectura de NASA-TLX Raw, donde el visor 3D también presentó menor carga de trabajo percibida.

El SUS se aplicó únicamente al prototipo 3D, de acuerdo con el diseño metodológico. El puntaje promedio fue 91,88, con mediana de 95,00, mínimo de 60,00, máximo de 100,00 y desviación estándar de 11,24. Frente a la referencia histórica de 68 puntos del instrumento, la distribución observada indica una percepción de usabilidad muy favorable, aunque se interpreta de forma descriptiva por el tamaño y carácter no probabilístico de la muestra (véanse la Tabla 24, la Tabla 25 y la Figura 76).

Tabla 24*Resultados SUS y NASA-TLX Raw por participante*

Código	Sec.	Dispositivo	SUS	NASA 3D	NASA 2D	Dif.	Lectura
001	AB	Smartphone	90,00	7,00	17,50	10,50	Usabilidad alta y menor carga en visor 3D.
002	BA	Smartphone	95,00	5,17	17,50	12,33	Usabilidad alta y baja carga 3D.
003	AB	Smartphone	97,50	8,67	22,83	14,16	Diferencia favorable al visor.
004	BA	PC	90,00	11,17	22,50	11,33	Flujo claro también en escritorio.
005	AB	Smartphone	90,00	7,50	24,17	16,67	Carga 2D más alta en comparación.
006	BA	Smartphone	60,00	25,00	36,67	11,67	Caso con mayor fricción, pero menor carga 3D.
007	AB	Smartphone	100,00	5,83	15,00	9,17	Experiencia muy favorable.
008	BA	Smartphone	100,00	10,83	16,67	5,84	Diferencia favorable moderada.
009	AB	Smartphone	100,00	9,17	15,00	5,83	Usabilidad máxima registrada.
010	BA	Smartphone	95,00	3,50	15,00	11,50	Menor carga 3D del grupo.
011	AB	Smartphone	85,00	5,17	16,67	11,50	Resultado favorable con fricciones menores.
012	BA	PC	100,00	5,33	19,17	13,84	Uso fluido en PC.

Nota. Sec. = secuencia de exposición; AB = visor 3D primero y soporte 2D después; BA = soporte 2D primero y visor 3D después. La diferencia se calcula como NASA-TLX Raw 2D menos NASA-TLX Raw 3D. Elaboración propia a partir de los instrumentos de validación aplicados.

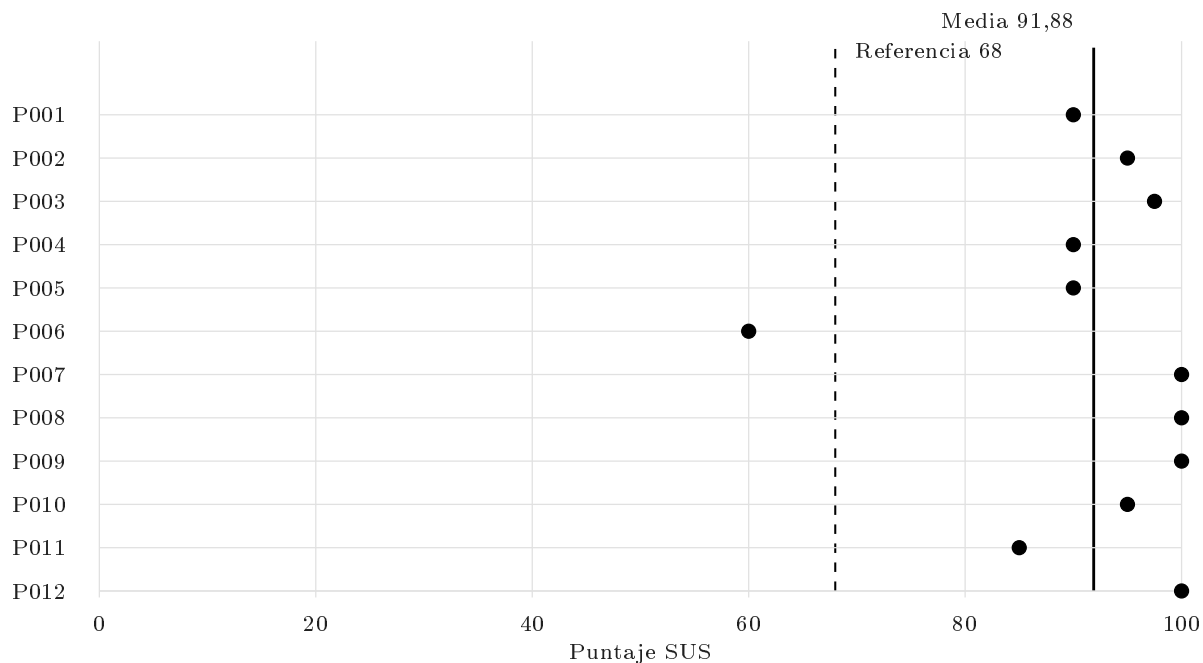
Tabla 25*Resumen descriptivo de usabilidad y carga de trabajo percibida*

Medida	n	Media	Mediana	Mín.	Máx.	DE
SUS	12	91,88	95,00	60,00	100,00	11,24
NASA-TLX Raw visor 3D	12	8,69	7,25	3,50	25,00	5,66
NASA-TLX Raw soporte 2D	12	19,89	17,50	15,00	36,67	6,18
Diferencia 2D–3D pareada	12	11,19	11,50	5,83	16,67	3,16

Nota. DE = desviación estándar. Los resultados se interpretan de forma descriptiva y exploratoria por el tamaño de muestra y el muestreo no probabilístico. Elaboración propia.

Figura 76

Distribución del puntaje SUS en la muestra evaluada.



Nota. La línea de referencia marca 68 puntos, valor usado como promedio histórico contextual del instrumento. La lectura del proyecto es descriptiva y no inferencial. Elaboración propia.

Resultados de carga de trabajo percibida

La lectura de NASA-TLX Raw confirma una menor carga de trabajo percibida en el visor 3D frente al soporte 2D de referencia. El promedio del visor fue 8,69, mientras que el soporte 2D obtuvo 19,89. La diferencia pareada media fue de 11,19 puntos a favor del visor 3D, y en los 12 casos el valor NASA-TLX Raw fue menor en la condición interactiva. Esta diferencia no se interpreta como prueba inferencial, sino como tendencia descriptiva consistente dentro de la muestra evaluada.

Por dimensiones, el visor 3D presentó menores promedios en demanda mental, demanda física, rendimiento invertido, esfuerzo y frustración; la demanda temporal fue equivalente entre condiciones. Las diferencias más visibles aparecieron en frustración, rendimiento percibido y demanda mental, lo que sugiere que la navegación 3D, la selección contextual y las vistas analíticas redujeron parte del esfuerzo de ubicación espacial requerido por el soporte 2D (véanse la Tabla 26 y la Figura 77).

Tabla 26

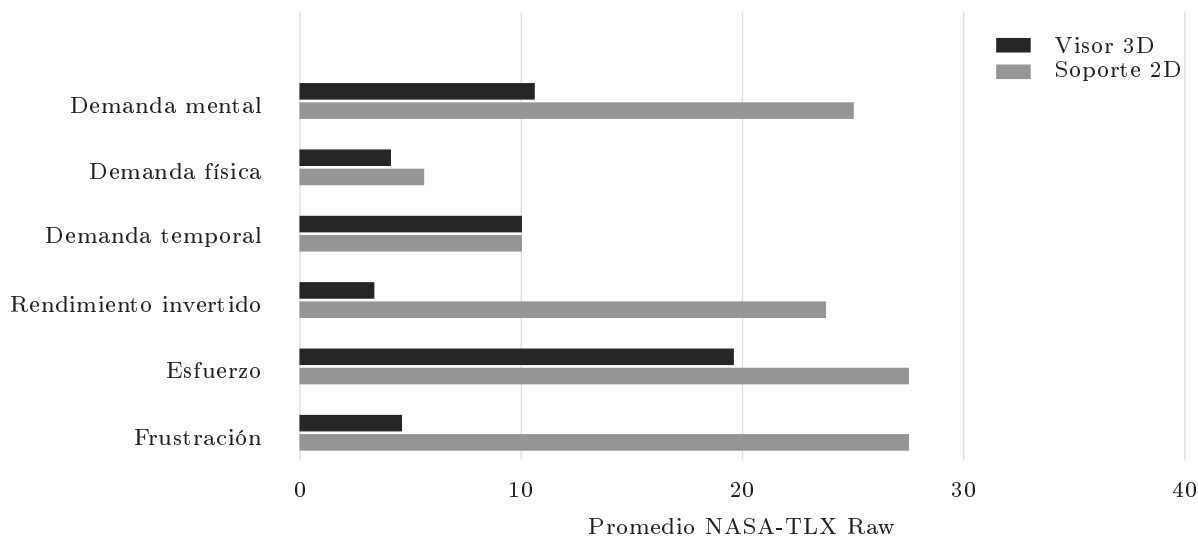
Promedios NASA-TLX Raw por dimensión y condición

Dimensión	Visor 3D	Soporte 2D	Diferencia
Demanda mental	10,58	25,00	14,42
Demanda física	4,08	5,58	1,50
Demanda temporal	10,00	10,00	0,00
Rendimiento invertido	3,33	23,75	20,42
Esfuerzo	19,58	27,50	7,92
Frustración	4,58	27,50	22,92

Nota. La dimensión de rendimiento se registró con escala invertida: 0 = perfecto y 100 = fallido. Por ello, valores menores indican mejor percepción de desempeño. Elaboración propia.

Figura 77

Comparativa NASA-TLX Raw por dimensiones.



Nota. La figura compara los promedios por dimensión para el visor 3D y el soporte 2D de referencia. Valores menores representan menor carga de trabajo percibida. Elaboración propia.

La convergencia entre menor NASA-TLX Raw, menor tiempo medio en T1–T3 y verbalizaciones orientadas a comprensión espacial sugiere que el valor del visor no se concentró en una sola métrica. La evidencia apunta a una experiencia en la que la interacción 3D ayudó a localizar, relacionar y consultar componentes con menor esfuerzo percibido, aunque la lectura se mantiene descriptiva por el tamaño y tipo de muestra.

Resultados cualitativos

El protocolo *Think-Aloud* permitió triangular los puntajes cuantitativos con verbalizaciones, observación de tareas y comentarios posteriores. La categoría más recurrente fue comprensión espacial, presente en 11 de los 12 participantes, seguida de navegación y control en 10 par-

ticipantes. Esto es coherente con la naturaleza del prototipo: los usuarios valoraron la posibilidad de entender relaciones entre motor, montura, tornillería y modos de visualización, pero también registraron fricciones propias de la interacción táctil y de la selección de piezas pequeñas.

Los temas de mejora más repetidos fueron inconsistencias visuales o estados incorrectos en iconos procedurales (6 participantes), problemas ocasionales de navegación móvil con órbita o *pan* (5 participantes) y dificultad para seleccionar piezas pequeñas o parcialmente ocultas (2 participantes). En contraste, los momentos de claridad se asociaron con el *bottom sheet*, el aislamiento, la vista explosionada, *X-Ray*, *Blueprint* y la visualización térmica. La Tabla 27 resume la codificación cualitativa y la Figura 78 la representa de forma visual.

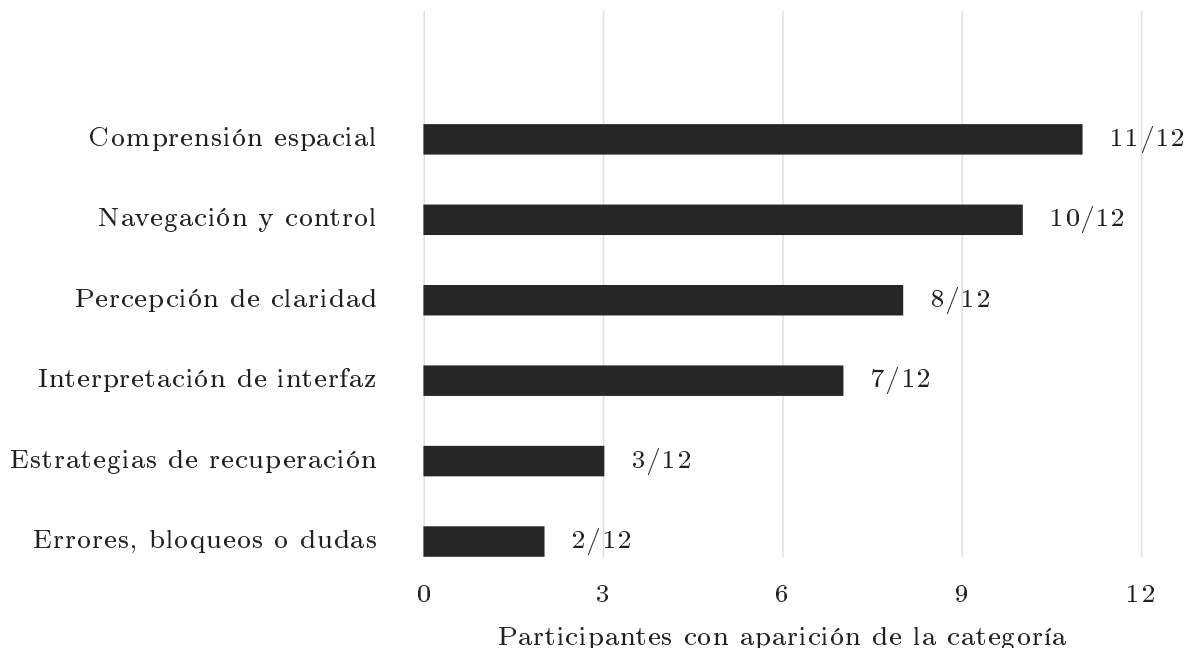
Tabla 27*Síntesis de hallazgos Think-Aloud*

Categoría analítica	Frecuencia	Evidencia representativa	Lectura
Comprensión espacial	11/12	Participantes usaron el visor para relacionar motor, montura, tornillería y disposición del conjunto.	Fortaleza principal del prototipo.
Navegación y control	10/12	En móvil se repitieron fricciones con órbita, <i>pan</i> y sensibilidad táctil.	Punto de mejora UX móvil.
Percepción de claridad	8/12	Vista explosionada, vista térmica, <i>X-Ray</i> , <i>Blueprint</i> y aislamiento facilitaron la lectura técnica.	Apoya la reducción de carga percibida.
Interpretación de interfaz	7/12	La lectura de iconos, menús y estados visuales concentró dudas y comentarios de mejora.	Requiere pulido visual continuo.
Estrategias de recuperación	3/12	Algunos usuarios cambiaron de ángulo, repitieron gestos o usaron capas de navegación para resolver dudas.	Indica aprendizaje durante la sesión.
Errores, bloqueos o dudas	2/12	La selección de piezas pequeñas o parcialmente ocultas generó fricción en casos específicos.	Riesgo localizado en elementos pequeños.

Nota. La frecuencia indica cantidad de participantes en los que apareció la categoría al menos una vez. Las verbalizaciones se sintetizan de forma anonimizada para preservar solo la información necesaria para el análisis. Elaboración propia.

Figura 78

Mapa de codificación cualitativa del protocolo Think-Aloud.



Nota. El mapa sintetiza frecuencia por categoría y lectura interpretativa de las verbalizaciones y observaciones registradas durante la prueba. Elaboración propia.

Resultados del pipeline de activos

La comparación de rutas de importación y saneamiento geométrico muestra que el problema principal no era solo reducir triángulos, sino transformar una malla CAD difícil de editar en un activo estable, legible y viable para WebGL. La Tabla 28 resume el conteo total de triángulos observado por ruta técnica. La Figura 79 complementa esta lectura con evidencia visual de la malla optimizada y su wireframe; la Figura 80 muestra el resultado visual del modelo final con mapas de normales y oclusión ambiental.

Tabla 28*Conteo total de triángulos por método de importación y optimización*

Método	Triángulos totales	Detalle técnico observado
Importación directa	6 717 499 tris	Malla caótica, conexiones en T, vértices repetidos, vértices no soldados y caras repetidas.
MoI3D	6 864 586 tris	Malla más limpia en lectura general, pero con n-gons, conexiones en T, vértices no soldados, vértices repetidos y caras repetidas.
STEPper	6 525 748 tris	Menor cantidad de vértices repetidos y caras repetidas frente a las rutas anteriores, pero con conexiones en T y malla desordenada en triángulos.
Modelo base optimizado	95 617 tris	Conteo correspondiente al activo base optimizado y a sus masters principales. No incluye proxies, instancias runtime, assets de apoyo ni renderers adicionales de la escena instrumentada.

Nota. La cifra de 95 617 triángulos corresponde al modelo base optimizado exportado para la build. No equivale al conteo de escena runtime reportado por el profiler, que puede ser mayor al incorporar instancias, proxies, assets de apoyo, renderers activos y otros recursos necesarios para la interacción. Elaboración propia.

Figura 79

Malla optimizada final y lectura de wireframe.



Nota. La comparación muestra el modelo optimizado y su equivalente en wireframe. Esta evidencia se ubica junto al conteo del activo base optimizado de 95 617 triángulos para relacionar reducción poligonal, limpieza topológica y legibilidad visual del activo. Elaboración propia.

Figura 80

Renders del dron optimizado con mapas de normales y oclusión ambiental.



Nota. Los renders corresponden al modelo low poly optimizado y permiten verificar que la reducción geométrica conserva una lectura formal adecuada mediante apoyo de mapas de normales y AO. Elaboración propia.

Nota de cierre

Los resultados muestran que el prototipo alcanzó un cierre funcional verificable, una reducción geométrica sustancial, rendimiento defendible para el alcance del MVP y una recepción favorable en la evaluación con usuarios. En términos cuantitativos, el SUS promedio fue alto, la carga NASA-TLX Raw fue menor para el visor 3D que para el soporte 2D en los 12 casos registrados y las cuatro tareas equivalentes se completaron en ambas condiciones. En términos cualitativos, los participantes asociaron el mayor valor del sistema con la comprensión espacial del ensamblaje, aunque persisten oportunidades de mejora en navegación móvil, lectura de iconos y selección de elementos pequeños.

Discusión

Los resultados muestran que el prototipo 3D no produjo una ventaja binaria de completitud frente al soporte 2D, porque las cuatro tareas se completaron en ambas condiciones y la matriz de desempeño presentó un efecto techo. Sin embargo, la comparación no quedó vacía: el visor 3D redujo de forma consistente la carga de trabajo percibida y, en las tres tareas cronometradas, presentó menores tiempos medios que la referencia 2D. La evidencia más fuerte del estudio no es una mayor tasa de éxito, sino una interacción espacial menos demandante y más eficiente en términos descriptivos dentro de la muestra evaluada.

Desde la Teoría de la Carga Cognitiva, este patrón puede leerse como una reducción de carga extrínseca más que como un aumento de capacidad intrínseca de resolución: el visor 3D no simplifica el sistema técnico en sí, pero sí externaliza relaciones espaciales que en el soporte 2D exigen reconstrucción mental adicional (Sweller et al., 2019; Hegarty & Waller, 2004). En esa medida, el hallazgo principal del estudio no es una superioridad absoluta en completitud, sino un

apoyo cognitivo más eficiente para tareas de inspección espacial.

Esta lectura es coherente con los resultados de NASA-TLX Raw. El visor 3D obtuvo un promedio de 8,69 frente a 19,89 del soporte 2D y fue menor en los 12 casos observados. Por dimensiones, las mayores diferencias aparecieron en demanda mental, rendimiento invertido y frustración, mientras que la demanda temporal fue equivalente entre condiciones. En conjunto, esto sugiere que la representación interactiva, la selección directa de piezas, el *bottom sheet*, el aislamiento, Explode, Cut y los modos visuales redujeron parte del esfuerzo de reconstrucción espacial que el soporte 2D exige al usuario.

Los tiempos descriptivos refuerzan esa interpretación. En T1, T2 y T3, el visor 3D registró medias de 5,75 s, 3,50 s y 11,33 s, frente a 13,00 s, 18,00 s y 23,00 s del soporte 2D. El tiempo total medio acumulado fue de 20,58 s en 3D y de 54,00 s en 2D. Estos datos deben leerse con cautela, porque la muestra fue no probabilística, el diseño fue formativo y no se aplicó estadística inferencial; aun así, complementan de modo consistente la lectura de menor carga percibida en la condición interactiva.

Desde el punto de vista técnico, la build mostró viabilidad clara en escritorio y comportamiento funcional en los dispositivos móviles evaluados, aunque no convirtió la compatibilidad móvil en una garantía universal. La evidencia por dispositivo muestra escritorio estable, comportamiento cercano a la meta en gama media-alta y restricciones en el límite inferior móvil. Del mismo modo, la reducción a 95 617 triángulos debe interpretarse como métrica del modelo base optimizado y no como conteo total de escena en runtime, que puede ser mayor por la presencia de instancias, proxies, assets de apoyo y renderers activos.

Estas observaciones delimitan con precisión el alcance académico del artefacto. Twin-Sight X500 se sostiene como *visual product twin* o modelo visual-semántico de producto: orga-

niza piezas, categorías, vistas analíticas y soporte contextual útil para inspección técnica, pero no integra telemetría en vivo, sincronización con activo físico, *twin manifest* formal ni simulación física calibrada. En consecuencia, el aporte principal de esta tesis no es cerrar un gemelo digital operacional, sino demostrar una vía técnicamente viable y metodológicamente defendible para traducir hardware complejo a una experiencia WebGL usable, interpretable y preparada para madurez futura.

Conclusiones

El proyecto permitió construir y validar de forma descriptiva TwinSight X500 como prototipo WebGL para visualización técnica de hardware complejo. La build publicada integra el flujo principal de exploración, selección, aislamiento, ficha contextual, modos de inspección/análisis/estudio, corte transversal, vista explosionada, filtros y visualización térmica heurística. Los resultados técnicos y de usuario se integraron con datos fechados: mediciones del profiler interno, reducción geométrica del activo y evaluación con 12 participantes mediante SUS, NASA-TLX Raw y protocolo *Think-Aloud*. Por su alcance real, el artefacto queda delimitado como *visual product twin*: un modelo visual-semántico de producto preparado para inspección técnica y evolución futura, sin telemetría en vivo ni sincronización operacional con un activo físico.

Respecto al objetivo general

Se logró construir un visor web interactivo para la visualización técnica del Holybro X500 V2, accesible desde navegador y capaz de integrar navegación 3D, selección de piezas, información contextual y herramientas de análisis visual. La validación formativa con 12 participantes registró un puntaje SUS promedio de 91,88, un promedio NASA-TLX Raw de 8,69 para el visor 3D frente a 19,89 para el soporte 2D y menor tiempo medio en T1, T2 y T3 para la condición 3D. Estos resultados se interpretan de forma descriptiva, dentro de una muestra no probabilística y un diseño formativo, y muestran una recepción favorable del prototipo junto con una menor carga de trabajo percibida durante la exploración 3D. Además, la taxonomía por piezas, los grupos de escena, las fichas y las herramientas de inspección constituyen una base razonable para evolucionar hacia modelos con datos operativos, siempre que esa evolución se aborde por

fases.

Respecto a los objetivos específicos

OE1: pipeline de optimización y preparación de activos. Se consolidaron el caso de estudio, la taxonomía canónica y un pipeline defendible que combina importación CAD, saneamiento geométrico, remodelado hard-surface, modularización delimitada de fasteners, automatización puntual y preparación para runtime web. La comparación geométrica muestra una reducción desde rutas CAD superiores a 6,5 millones de triángulos hasta un modelo base optimizado de 95 617 triángulos, incluyendo piezas adicionales modeladas desde cero como hélices. Esta reducción permitió conservar legibilidad formal y hacer viable la interacción WebGL dentro del alcance del MVP, sin confundir ese conteo de activo base con la complejidad de la escena runtime instrumentada.

OE2: materiales, modos de visualización y rendimiento WebGL. Se integró en Unity URP una capa visual que combina renderizado realista, modos analíticos y presets de presentación: *Realistic*, *X-Ray*, *Solid View*, *Thermal*, *Blueprint*, entornos de *Studio*, corte transversal, filtros y vista explosionada. Las mediciones muestran cumplimiento pleno de la meta de 30 FPS en escritorio y equipos de mayor capacidad; en móviles de gama media-alta el comportamiento fue cercano a la meta y en el límite inferior quedó por debajo de 30 FPS, aunque mantuvo navegabilidad en el flujo principal. Por ello, la compatibilidad móvil debe leerse como funcional en los dispositivos evaluados y no como garantía universal para cualquier hardware.

OE3: implementación del prototipo interactivo. Se obtuvo una arquitectura funcional basada en Unity Web, C#, UI Toolkit, gestores de escena, orquestación de modos y saneamiento del import mediante `ImportedDroneRuntimeBinder`. La build publicada integra navegación, selección, aislamiento, ficha contextual inferior, menús *Inspect*, *Analyze* y *Studio*,

Explode, Cut, filtros, modos visuales, tutorial y adaptación de la experiencia *mobile-first* a escritorio. La separación entre UI visible, código oculto y código no integrado mejora la trazabilidad del sistema y evita documentar como alcance final capacidades que permanecen fuera del flujo publicado.

OE4: evaluación formativa 3D vs. 2D, usabilidad y carga de trabajo. La evaluación con 12 participantes permitió describir la experiencia del prototipo desde capas complementarias. La matriz de desempeño registró cuatro tareas equivalentes por condición y 96 registros tarea-condición completados, con una ayuda baja en el visor 3D y sin errores de tarea en la matriz estructurada. En las tres tareas cronometradas, el visor 3D presentó menor tiempo medio que el soporte 2D, mientras que NASA-TLX Raw mostró menor carga percibida en la condición 3D en los 12 casos observados. La completitud total en ambas condiciones sugiere un efecto techo en la matriz de desempeño, por lo que la ventaja del visor 3D se interpreta principalmente en términos de menor demanda subjetiva, mejor orientación espacial y menor tiempo descriptivo de resolución, no como prueba inferencial de superioridad causal. El *Think-Aloud* explicó esta lectura: los participantes valoraron la comprensión espacial, el aislamiento, las vistas analíticas y el panel de pieza, mientras que las principales fricciones se concentraron en navegación móvil, lectura de iconos y selección de elementos pequeños. La documentación producida queda como soporte de transferencia y mantenimiento, no como objetivo específico independiente.

Limitaciones

- la muestra de usuarios fue no probabilística y estuvo compuesta por 12 participantes, por lo que los resultados se interpretan de manera descriptiva y formativa;
- las observaciones sobre rendimiento corresponden a los dispositivos, navegadores y sesiones registradas; no constituyen garantía universal para todos los equipos o condi-

- ciones de red;
- las sesiones con usuarios se realizaron en dispositivos controlados y sobre la misma build trazada en la rama pública *master*, la URL publicada y los artefactos versionados de cierre. La caché del navegador se cargó previamente, lo que mejora la trazabilidad de la medición pero limita la generalización a otros equipos;
 - la T4 se trató como tarea exploratoria guiada y no como tarea cronometrada, por lo que su aporte se interpreta desde la observación cualitativa y no desde el análisis temporal;
 - los tiempos de T1–T3 se reportan como evidencia descriptiva de desempeño, sin inferencia estadística por el tamaño y tipo de muestra;
 - la visualización térmica debe entenderse como simulación híbrida heurística por componentes, no como FEA ni medición física calibrada;
 - el sistema no integra telemetría real, historial operacional, conectores IoT, SCADA, CMMS o PLM ni sincronización con activo físico;
 - no existe todavía un *twin manifest* formal que convierta cada pieza en entidad interoperable con sensores, fallas, repuestos y procedimientos;
 - varias herramientas existen en código pero no hacen parte de la UI final visible;
 - no se incluyeron cables, conexiones ni todos los elementos electrónicos internos por restricciones de tiempo de modelado, optimización e interacción dentro de Unity;
 - la versión de escritorio funciona correctamente como adaptación del diseño *mobile-first*, aunque no constituye un rediseño desktop independiente con el mismo nivel de profundidad.

Trabajo futuro

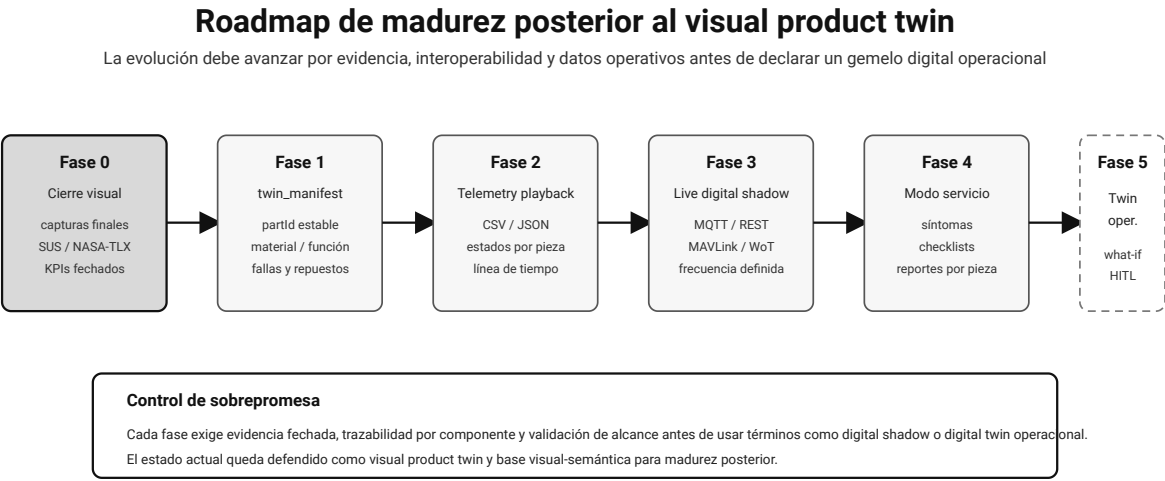
El trabajo futuro se organiza como una ruta de madurez incremental (véase la Figura 81). La prioridad consiste en ampliar evidencia, datos e interoperabilidad por etapas, antes de aspirar a inteligencia artificial predictiva o a un gemelo digital operacional completo.

1. **Fase 0: ampliación del visual product twin.** Replicar la validación con una muestra mayor, idealmente cercana a 30 participantes, contrastar nuevos dispositivos y depurar la interacción móvil con énfasis en navegación, iconos y selección de piezas pequeñas.
2. **Fase 1: formalización del *twin manifest*.** Definir un esquema JSON o equivalente que conecte cada pieza con identificador estable, categoría, material, función, sensores posibles, variables esperadas, fallas comunes, repuestos, procedimientos y documentos asociados.
3. **Fase 2: telemetría histórica y digital shadow de reproducción.** Importar logs o datasets CSV/JSON para reproducir estados por componente en una línea de tiempo, sin exigir todavía conexión en vivo.
4. **Fase 3: live digital shadow.** Incorporar adaptadores WebSocket, MQTT, REST, MAVLink, ROS, OPC UA o W3C WoT según el dominio, con frecuencia de actualización definida y control de rendimiento WebGL.
5. **Fase 4: modo servicio y mantenimiento guiado.** Añadir síntomas, causas probables, checklists, herramientas, repuestos, evidencia, severidad y exportación de reportes por pieza.
6. **Fase 5: digital twin operacional.** Integrar modelos calibrados, simulaciones *what-*

if, predicción de umbrales y decisiones human-in-the-loop solo cuando existan datos históricos confiables y validación con eventos reales.

- 7. **Mejoras transversales.** Diseñar una UI desktop específica, modelar cables y conexiones electrónicas relevantes, evaluar con datos si una ruta nativa de LOD aporta una mejora real al pipeline, calibrar el subsistema térmico e incorporar soporte multi-idioma.

Figura 81
Roadmap visual de evolución desde visual product twin hasta digital twin operacional.



Nota. La ruta separa el cierre visual documentado de fases posteriores con manifest, telemetría, sombra digital en vivo, modo servicio y gemelo operacional. Elaboración propia.

Recomendaciones

- 1. mantener una única taxonomía documental para piezas, *anchors* y *renderers*;
- 2. documentar como finales solo las funciones visibles y verificables de la build publica-

- da;
3. preservar trazabilidad de versión, dispositivo, navegador y fecha cada vez que se actualicen métricas técnicas o pruebas de usuario;
 4. preservar el resto del código como base técnica, distinguiendo siempre entre integrado, oculto y futuro;
 5. utilizar “visual product twin” para describir el alcance actual y reservar “digital shadow” o “digital twin” para fases con datos operativos, sincronización y validación.

Referencias

- Akenine-Möller, T., Haines, E., & Hoffman, N. (2018). *Real-time rendering* (4th ed.). CRC Press.
- Bangor, A., Kortum, P. T., & Miller, J. T. (2008). An empirical evaluation of the system usability scale. *International Journal of Human-Computer Interaction*, 24(6), 574–594. <https://doi.org/10.1080/10447310802205776>
- Bangor, A., Kortum, P. T., & Miller, J. T. (2009). Determining what individual SUS scores mean: Adding an adjective rating scale. *Journal of Usability Studies*, 4(3), 114–123.
- Babylon.js. (s. f.). *Engine specifications*. <https://www.babylonjs.com/specifications/>
- Bartlett, K. A., & Dorribo Camba, J. (2023). The role of a graphical interpretation factor in the assessment of spatial visualization: A critical analysis. *Spatial Cognition & Computation*, 23(1), 1–30. <https://doi.org/10.1080/13875868.2021.2019260>
- Blender Bros. (2022). *The Blender Bros hard surface game asset course 2.0*. <https://www.blenderbros.com/gameasset>
- Bowman, D. A., Kruijff, E., LaViola, J. J., Jr., & Poupyrev, I. (2004). *3D user interfaces: Theory and practice*. Addison-Wesley.
- Brooke, J. (1996). SUS: A “quick and dirty” usability scale. En P. W. Jordan, B. Thomas, I. L. McClelland, & B. Weerdmeester (Eds.), *Usability evaluation in industry* (pp. 189–194). Taylor & Francis.
- Burley, B. (2012). Physically-based shading at Disney. *ACM SIGGRAPH 2012 course notes: Practical physically based shading in film and game production*. ACM.
- Cabello, R. [mrdoob]. (2024). *three.js – JavaScript 3D library* [Repositorio de GitHub]. GitHub.

<https://github.com/mrdoob/three.js>

- Cook, R. L., & Torrance, K. E. (1982). A reflectance model for computer graphics. *ACM Transactions on Graphics*, 1(1), 7–24. <https://doi.org/10.1145/357290.357293>
- Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, 24(1), 87–114.
- Darken, R. P., & Sibert, J. L. (1996). Wayfinding strategies and behaviours in large virtual worlds. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 142–149. <https://doi.org/10.1145/238386.238459>
- David, I., Shao, G., Tilbury, D., Gomes, C., & Zarkhout, B. (2024). *Interoperability of digital twins: Challenges, success factors, and future research directions*. National Institute of Standards and Technology. <https://www.nist.gov/publications/interoperability-digital-twins-challenges-success-factors-and-future-research>
- Digital Twin Consortium. (2020, 3 de diciembre). *Digital Twin Consortium defines digital twin*. <https://www.digitaltwinconsortium.org/2020/12/digital-twin-consortium-defines-digital-twin/>
- Ericsson, K. A., & Simon, H. A. (1993). *Protocol analysis: Verbal reports as data* (Revised ed.). MIT Press.
- Epic Games. (s. f.-a). *Unreal Engine Pixel Streaming Reference*. Epic Developer Community. https://dev.epicgames.com/documentation/en-us/unreal-engine/unreal-engine-pixel-streaming-reference?application_version=5.6

- Epic Games. (s. f.-b). *Pixel Streaming Infrastructure*. Epic Developer Community. https://dev.epicgames.com/documentation/en-us/unreal-engine/pixel-streaming-infrastructure?application_version=5.6
- Fransson, E., Hermansson, J., & Hu, Y. (2024). A comparison of performance on WebGPU and WebGL in the Godot game engine. En *2024 IEEE Gaming, Entertainment, and Media Conference (GEM)* (pp. 1–6). IEEE. <https://doi.org/10.1109/GEM61861.2024.10585437>
- Fuller, A., Fan, Z., Day, C., & Barlow, C. (2020). Digital twin: Enabling technologies, challenges and open research. *IEEE Access*, 8, 108952–108971. <https://doi.org/10.1109/ACCESS.2020.2998358>
- Hart, S. G. (2006). NASA-task load index (NASA-TLX): 20 years later. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 50(9), 904–908. <https://doi.org/10.1177/154193120605000909>
- Hart, S. G., & Staveland, L. E. (1988). Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. En P. A. Hancock & N. Meshkati (Eds.), *Human mental workload* (pp. 139–183). North-Holland.
- Hegarty, M., & Waller, D. (2004). Spatial abilities at different scales: Individual differences in aptitude-test performance and spatial-layout learning. *Intelligence*, 32(2), 151–176.
- Hevner, A. R., March, S. T., Park, J., & Ram, S. (2004). Design science in information systems research. *MIS Quarterly*, 28(1), 75–105.
- Holybro. (s. f.-a). *PX4 development kit – X500v2*. Holybro Docs. <https://docs.holybro.com/drone-development-kit/px4-development-kit-x500v2>
- Holybro. (s. f.-b). *Spare parts – X500 V2 kit*. Holybro Store. <https://holybro.com/prod>

ucts/spare-parts-x500-v2-kit

Holybro. (s.f.-c). *Holybro X500 V2 frame kit assembly guide* [Guía del fabricante]. https://cdn.shopify.com/s/files/1/0604/5905/7341/files/Holybro_X500_V2_Frame_Kit_Assembly_Guide_en.pdf?v=1646987799

Hutchins, E. (1995). *Cognition in the wild*. MIT Press.

International Organization for Standardization. (2021). *ISO 23247-1:2021 Automation systems and integration – Digital twin framework for manufacturing – Part 1: Overview and general principles*. ISO. <https://www.iso.org/standard/75066.html>

Jones, D., Snider, C., Nassehi, A., Yon, J., & Hicks, B. (2020). Characterising the digital twin: A systematic literature review. *CIRP Journal of Manufacturing Science and Technology*, 29, 36–52. <https://doi.org/10.1016/j.cirpj.2020.02.002>

Kajiya, J. T. (1986). The rendering equation. *ACM SIGGRAPH Computer Graphics*, 20(4), 143–150. <https://doi.org/10.1145/15886.15902>

Karis, B. (2013). Real shading in Unreal Engine 4. En B. Burley (Ed.), *SIGGRAPH 2013 course: Physically based shading in theory and practice*. ACM.

Khronos Group. (2011). *WebGL Specification 1.0*. <https://www.khronos.org/registry/webgl/specs/latest/1.0/>

Khronos Group. (2024). *glTF – Runtime 3D asset delivery*. <https://www.khronos.org/glTF/>

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital twin in manufacturing: A categorical literature review and classification. *IFAC-PapersOnLine*, 51(11), 1016–1022. <https://doi.org/10.1016/j.ifacol.2018.08.474>

Köhler, W. (1929). *Gestalt psychology*. Liveright.

- Lazar, J., Feng, J. H., & Hochheiser, H. (2017). *Research methods in human-computer interaction* (2nd ed.). Morgan Kaufmann.
- Lin, S.-W., Watson, K., Shao, G., Stojanovic, L., & Zarkout, B. (2023). *Digital twin core conceptual models and services*. National Institute of Standards and Technology. <https://www.nist.gov/publications/digital-twin-core-conceptual-models-and-services>
- Miller, G. A. (1956). The magical number seven, plus or minus two: Some limits on our capacity for processing information. *Psychological Review*, 63(2), 81–97.
- Marmoset. (s. f.). *Marmoset Viewer (.mview)*. Marmoset Toolbag Documentation. <https://docs.marmoset.co/docs/marmoset-viewer/>
- Nielsen, J. (1994). *Usability engineering*. Morgan Kaufmann.
- Nielsen, J. (2000). Why you only need to test with 5 users. Nielsen Norman Group. <https://www.nngroup.com/articles/why-you-only-need-to-test-with-5-users/>
- Norman, D. A. (2013). *The design of everyday things: Revised and expanded edition*. Basic Books.
- Open Cascade. (s. f.). *glTF model assets*. OCCT3D Documentation. https://occt3d.com/doc/ocviewer/3.0.354/documents/Guide.Model_types.glTF.html
- Paivio, A. (1986). *Mental representations: A dual coding approach*. Oxford University Press.
- Peffers, K., Tuunanen, T., Rothenberger, M. A., & Chatterjee, S. (2007). A design science research methodology for information systems research. *Journal of Management Information Systems*, 24(3), 45–77. <https://doi.org/10.2753/MIS0742-122224030>

- PlayCanvas. (s.f.-a). *Editor*. <https://playcanvas.com/products/editor>
- PlayCanvas. (s.f.-b). *Engine*. <https://playcanvas.com/products/engine>
- PlayCanvas. (s.f.-c). *Scripting*. <https://developer.playcanvas.com/user-manual/scripting/>
- PX4 Development Team. (2026). *Simulation*. PX4 User Guide. <https://docs.px4.io/main/en/simulation/>
- Sauro, J., & Lewis, J. R. (2016). *Quantifying the user experience: Practical statistics for user research* (2nd ed.). Morgan Kaufmann.
- Schlick, C. (1994). An inexpensive BRDF model for physically-based rendering. *Computer Graphics Forum*, 13(3), 233–246.
- Sharma, A., Kosasih, E., Zhang, J., Brintrup, A., & Calinescu, A. (2022). Digital twins: State of the art theory and practice, challenges, and open research questions. *Journal of Industrial Information Integration*, 30, 100383. <https://doi.org/10.1016/j.jii.2022.100383>
- Shepard, R. N., & Metzler, J. (1971). Mental rotation of three-dimensional objects. *Science*, 171(3972), 701–703.
- Sketchfab. (s.f.). *Viewer API*. <https://sketchfab.com/developers/viewer>
- Spline. (s.f.-a). *Play mode*. <https://docs.spline.design/basics/play-mode>
- Spline. (s.f.-b). *Exporting as Spline Viewer*. <https://docs.spline.design/doc/exporting-as-spline-viewer/docoQp60n9ht>
- Spline. (s.f.-c). *Exporting as glTF/GLB*. <https://docs.spline.design/exporting-your-scene/files/exporting-as-gltf-glb>
- Sweller, J. (1988). Cognitive load during problem solving: Effects on learning. *Cognitive Science*,

12(2), 257–285.

Sweller, J., van Merriënboer, J. J. G., & Paas, F. (2019). Cognitive architecture and instructional design: 20 years later. *Educational Psychology Review*, 31(2), 261–292. <https://doi.org/10.1007/s10648-019-09465-5>

Tao, F., Zhang, H., Liu, A., & Nee, A. Y. C. (2019). Digital twin in industry: State-of-the-art.

IEEE Transactions on Industrial Informatics, 15(4), 2405–2415. <https://doi.org/10.1109/TII.2018.2873186>

Unity Technologies. (2024a). *Memory in WebGL*. Unity Documentation. <https://docs.unity3d.com/Manual/webgl-memory.html>

Unity Technologies. (2024b). *Universal Render Pipeline overview*. Unity Documentation. <https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@17.0/manual/>

Unity Technologies. (2024c). *WebGL: Interacting with browser scripting*. Unity Documentation. <https://docs.unity3d.com/Manual/webgl-interactingwithbrowser scripting.html>

Unity Technologies. (2024d). *UI Toolkit*. Unity Documentation. <https://docs.unity3d.com/Manual/UIElements.html>

Unity Technologies. (s.f.). *Web browser compatibility*. Unity Documentation. <https://docs.unity3d.com/6000.0/Manual/webgl-browsercompatibility.html>

Walter, B., Marschner, S. R., Li, H., & Torrance, K. E. (2007). Microfacet models for refraction through rough surfaces. *Proceedings of the 18th Eurographics Conference on Rendering Techniques*, 195–206.

Ware, C. (2012). *Information visualization: Perception for design* (3rd ed.). Morgan Kaufmann.

WebAssembly Community Group. (s. f.). *WebAssembly core specification*. <https://webassembly.github.io/spec/core/>

Wertheimer, M. (1923). Untersuchungen zur Lehre von der Gestalt II. *Psychologische Forschung*, 4(1), 301–350.

World Wide Web Consortium. (2023). *Web of Things (WoT) Architecture 1.1*. W3C Recommendation. <https://www.w3.org/TR/wot-architecture11/>

Yu, G., Liu, C., Fang, T., Jia, J., Lin, E., He, Y., Fu, S., Wang, L., Wei, L., & Huang, Q. (2023). A survey of real-time rendering on Web3D application. *Virtual Reality & Intelligent Hardware*, 5(5), 379–394. <https://doi.org/10.1016/j.vrih.2022.04.002>

Apéndices

Este capítulo lista los artefactos anexos activos que respaldan el informe, la app publicada, la validación técnica y la evaluación con usuarios.

Apéndice A: Repositorio de código fuente

Repositorio principal del proyecto:

```
https://github.com/delarge95/WebGL-Thesis-Proposal
```

Este repositorio, junto con el *snapshot* local del proyecto y los anexos versionados, funciona como referencia técnica principal para escena, scripts, herramientas de editor y documentación de desarrollo. La copia pública de cierre se consolida en la rama `master`; por tanto, la trazabilidad pública debe verificarse contra el `HEAD` vigente de esa rama, la URL de demo publicada y los artefactos activos listados en este capítulo.

Si el repositorio público no expone todos los artefactos internos por razones de limpieza, peso o información sensible, la evaluación académica debe apoyarse en la URL de demo publicada, los anexos activos del informe, la documentación técnica interna versionada y el *snapshot* local disponible para revisión.

Apéndice B: Manual de usuario

Ruta del manual activo:

```
Informe_final/Manual_de_usuario/manual_usuario.tex
```

Apéndice C: Manual técnico

Ruta del manual activo:

```
Informe_final/Manual_tecnico/manual_tecnico.tex
```


Apéndice D: Documentación técnica activa

Documentos activos:

- Informe_final/Desarrollo_App/Documentacion_Tecnica/01_Arquitectura_del_Sistema.md
- Informe_final/Desarrollo_App/Documentacion_Tecnica/02_Referencia_Tecnica_Modulos.md
- Informe_final/Desarrollo_App/Documentacion_Tecnica/03_Pipeline_Renderizado_Shaders.md
- Informe_final/Desarrollo_App/Documentacion_Tecnica/04_Guia_Despliegue.md
- Informe_final/Desarrollo_App/Documentacion_Tecnica/05_Configuracion_WebGL.md
- Informe_final/Desarrollo_App/Documentacion_Tecnica/08_Sistema_Termico_Hibrido.md

La versión publicada para consulta desde la landing se conserva en docs/manuals/. Esta copia se sincroniza con los documentos internos cuando se actualiza el cierre técnico.

Apéndice E: Matriz de desconexiones

La matriz maestra de desconexiones entre app y documentación se encuentra en:

Informe_final/Desarrollo_App/MATRIZ_DESCONEXIONES_APP_DOCUMENTACION.md

Apéndice F: Catálogo canónico del Holybro X500 V2

La taxonomía semántica del proyecto se fija a partir de:

desarrollo/unity_project/Assets/Resources/x500v2_info_panel_bilingual_

catalog.json

Esta fuente pública integra los identificadores canónicos, las fichas bilingües de información y las claves de trazabilidad utilizadas por el panel de la app. La taxonomía académica conserva 28 piezas canónicas; la escena final opera adicionalmente con 30 *anchors* y 257 *renderers/colliders* auditados.

Apéndice G: Validación funcional de la escena final

El documento de referencia para la build funcional es:

Informe_final/Desarrollo_App/VALIDACION_FUNCIONAL_FINA_2026-04-09.md

Apéndice H: Instrumentos y registros de evaluación

Los instrumentos metodológicos activos del cierre son:

- cuestionario SUS;
- NASA-TLX Raw;
- protocolo *Think-Aloud*;
- guía de mediciones técnicas WebGL;
- guía de tareas comparativas 3D vs. 2D;
- tablas de KPIs técnicos asociadas a sesiones fechadas del profiler interno.

Las tareas de prueba se alinean con cuatro acciones equivalentes: ubicar un componente, interpretar su función, explicar una relación espacial y resolver una acción de análisis visual guiado.

En la condición 3D estas tareas se apoyan en navegación, selección, *bottom sheet*, *Explode*, Cut y modos visuales; en la condición 2D se resuelven mediante el paquete documental estático de referencia.

Para mantener coherencia con la metodología vigente, SUS se reporta como lectura global del

prototipo 3D y NASA-TLX Raw se reporta como comparación de carga de trabajo percibida entre la condición 3D y el soporte 2D de referencia.

Tabla 29

Matriz de equivalencia informativa entre visor 3D y soporte 2D

Tarea	Condición 3D	Condición 2D
Ubicación de componente	Navegación orbit/pan/zoom, selección directa y foco sobre pieza.	Vista general, lámina de rotulado o página del soporte con el mismo componente.
Interpretación funcional	Ficha inferior de pieza, categoría y lectura contextual del ensamblaje.	Ficha textual o fragmento documental equivalente del paquete 2D.
Relación espacial	Inspección 3D, aislamiento, vista explosionada o corte cuando aplica.	Despiece, vista del frame o diagrama estático con relación entre piezas.
Análisis visual guiado	Uso de <i>Explode</i> , <i>Cut</i> , <i>X-Ray</i> , <i>Thermal</i> o presets de <i>Studio</i> .	Interpretación de vista estática equivalente para responder la misma pregunta técnica.

Nota. La equivalencia es funcional, no formal: el visor 3D ofrece interacción espacial, mientras que el soporte 2D conserva información suficiente para resolver la misma intención de tarea. Elaboración propia.

Las versiones activas y versionadas de instrumentos y registros se conservan en:

- Informe_final/validacion/CUESTIONARIO_SUS.md
- Informe_final/validacion/CUESTIONARIO_NASA_TLX.md
- Informe_final/validacion/GUIA_TAREAS_VALIDACION.md
- Informe_final/validacion/PROTOCOLO_THINK_ALOUD.md
- Informe_final/validacion/06_GUIA_MEDICIONES_TECNICAS_WEBGL.md
- Informe_final/validacion/07_TABLAS_RENDIMIENTO_WEBGL_MEDICIONES.tex

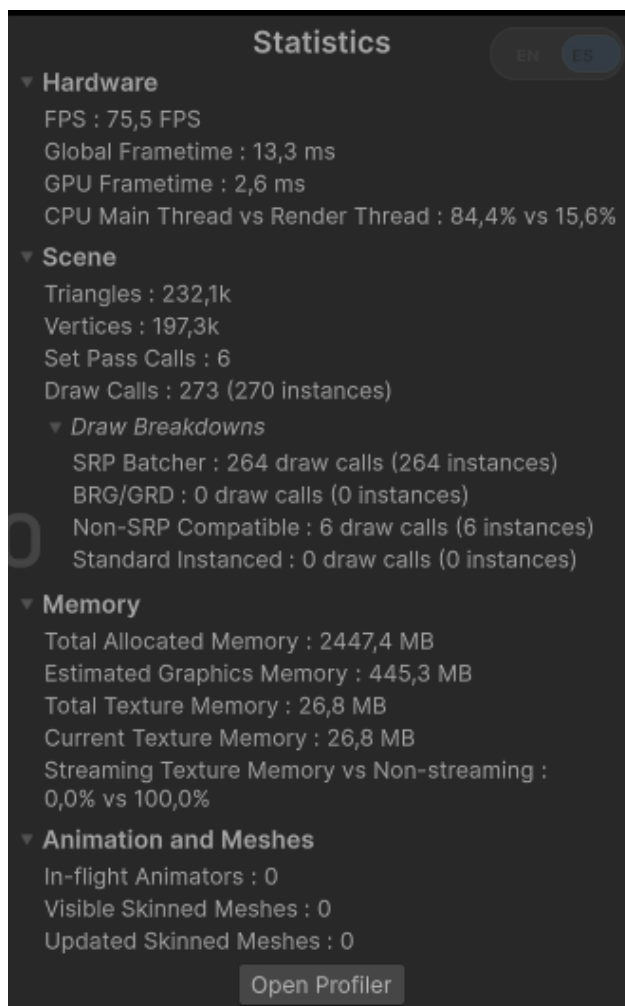
Los registros completos por participante se conservan como evidencia interna de evaluación y no forman parte de la superficie pública del repositorio. En el capítulo 5 se reportan únicamente datos agregados y participantes anonimizados; si un jurado o asesor requiere auditar los registros individuales, estos deben entregarse por un canal académico controlado y no mediante publica-

ción pública.

Como respaldo visual del proceso de medición técnica, la Figura 82 muestra una captura puntual del profiler interno conservada junto con las capturas de cierre de la aplicación.

Figura 82

Muestra de una captura del profiler interno.



Nota. La figura documenta una captura puntual del profiler interno integrada como evidencia visual en Informe_final/figures/screenshots_contextual/fig_profiler_internal_evidence.png. Su función es respaldar visualmente el proceso de recolección técnica; los valores oficiales de FPS, memoria, renderers, mallas y triángulos reportados en las tablas técnicas provienen de las exportaciones JSON/CSV del profiler interno, no de lectura manual de esta imagen. Elaboración propia.

Apéndice I: Estado del pipeline de activos

Para efectos de cierre documental:

- Blender y Unity conforman la ruta defendible de la versión final activa;
- el FBX runtime final se documenta como unión de masters e instancias; su trazabilidad exige reporte de importación Unity, revisión de grupos, fasteners, hotspots, filtros y hélices;
- cualquier ruta nativa de LOD se documenta como decisión evaluada; si existen herramientas o scripts posteriores de LOD, no se presentan como mejora validada del estudio mientras no exista profiling y QA comparativo sobre la build publicada;
- Houdini se mantiene como herramienta evaluada o experimental, no como pipeline oficial de la build cerrada, salvo evidencia nueva derivada del reimport final;
- las texturas/materiales no publicados deben mantenerse como mejora futura o documentarse expresamente si se integran a una build posterior.

El estado actual del pipeline puede verse consolidado en la Tabla 30.

Tabla 30*Estado del pipeline 3D al cierre documental*

Etapas	Herramienta principal	Estado	Lectura editorial
Preparación y limpieza de activos	Blender	Activa	Ruta defendible del cierre para ajuste de malla, organización y exportación.
Conversión CAD con control de teselación	Blender directo para inspección / STEPper / MoI3D	Activa / evaluada	El pipeline combinó rutas de apertura rápida y rutas con mayor control de teselación según la complejidad geométrica de cada pieza.
Texturizado y materiales	Unity URP / materiales del proyecto	Activa	El informe final documenta el flujo material realmente visible en la build y sus restricciones web.
Integración en motor	Unity	Activa	Núcleo del producto final y punto de verdad para escena, UI y <i>runtime binding</i> .
Auditoría de cobertura y <i>runtime repair</i>	Tooling de editor + binder de runtime importado	Activa	Parte esencial del cierre técnico y de la coherencia entre escena y documentación.
Ruta nativa de LOD	Investigación técnica del proyecto	Evaluada / no validada comparativamente	Debe explicarse como evolución de planeación técnica; si existen scripts posteriores, no deben presentarse como mejora medida de la build sin profiling y QA comparativo.
Houdini	Evaluación experimental	No canónico	Puede citarse como exploración o trabajo futuro, pero no como pipeline oficial de la versión final.

Nota. Esta tabla evita presentar como pipeline oficial herramientas que no quedaron integradas de forma cerrada en la build final. Elaboración propia.

Apéndice J: Demo público

La URL pública del demo publicado para la entrega es:

<https://delarge95.github.io/WebGL-Thesis-Proposal/>

Esta URL corresponde al punto de acceso a la build WebGL publicada. Las métricas del capítulo 5 registran la fecha y el contexto de medición utilizado para evitar inconsistencias entre documento y producto.

Apéndice K: Audio y alcance no integrado

Al momento de este cierre documental:

- no existen clips de audio integrados en *Assets*;
- por defecto, el audio se clasifica como trabajo futuro si no entra en la build publicada;

- el capítulo 5 reporta la evaluación técnica y de usuarios con los datos disponibles para esta versión del informe.

Apéndice L: Auditoría matemática y trazabilidad con Wolfram

La auditoría activa de coherencia entre fórmulas documentadas, lógica de código y arquitectura real se consolida en:

```
Informe_final/Desarrollo_App/AUDITORIA_MATEMATICA_Y_ARQUITECTURA_2026-0
4-12.md
```

Como soporte técnico adicional, el subsistema térmico se documenta en el anexo técnico público:

```
Informe_final/Desarrollo_App/Documentacion_Tecnica/08_Sistema_Termico_
Hibrido.md
```

Ambos documentos respaldan la interpretación correcta del modo *Thermal* como simulación híbrida heurística y no como FEA en tiempo real. Los registros históricos usados durante el desarrollo permanecen como evidencia interna cuando no forman parte de la superficie pública del repositorio.

Apéndice M: Correlación entre la propuesta y el informe final

La siguiente tabla resume la correlación de las tablas presentadas en la propuesta (véase la Tabla 31).

Tabla 31*Correlación entre tablas de la propuesta y el informe final*

Tabla en la propuesta	Estado en el informe final	Justificación
Comparativa de soluciones Web 3D	Conservada en el Capítulo 2	Sigue siendo pertinente como sustento del estado del arte y de la selección de Unity Web.
Cronograma general del proyecto	Se conserva solo en la propuesta	Es un artefacto de planeación; en el informe final pierde valor analítico y se sustituye por narrativa de ejecución y cierre.
Presupuesto estimado de recursos	Se conserva solo en la propuesta	Corresponde a la viabilidad ex ante del proyecto; no es un resultado ni un artefacto técnico del cierre.
Resultados o productos esperados	Transformada en tabla de productos entregados vs. esperados	En el informe final se requiere contrastar expectativa inicial con estado real de entrega y frentes de medición.

Nota. La ausencia de una tabla del documento final no implica incoherencia si su función era exclusivamente prospectiva en la propuesta. Elaboración propia.

Asimismo, la Tabla 32 presenta un contraste entre los productos entregados y los originalmente esperados.

Tabla 32*Productos entregados vs. esperados al cierre documental*

Producto esperado en la propuesta	Estado	Observación de cierre
Prototipo de software web funcional	Cerrado funcionalmente	El flujo principal es funcional, publicado y documentable; las métricas cuantitativas se consolidan por escenario y dispositivo.
Sistema de shaders URP optimizados	Entregado	La arquitectura y los modos visibles/ocultos quedaron documentados; la versión final debe seguir diferenciando lo publicado de lo oculto.
Conjunto de modelos 3D optimizados	Entregado para el alcance actual	Existe pipeline defendible y escena funcional; mejoras de reimport, texturas o fasteners Blender quedan como trabajo futuro si no forman parte de la build publicada.
Documento de trabajo de grado	En cierre final	Propuesta e informe final quedaron saneados; el capítulo 5 organiza datos reales de rendimiento y usuarios.
Informe de usabilidad y carga de trabajo percibida	Entregado con datos de validación	El capítulo 5 consolida SUS, NASA-TLX Raw, tareas completadas, observación <i>Think-Aloud</i> y lectura cualitativa por categorías.

Nota. Esta tabla actualiza el horizonte de productos de la propuesta sin borrar la diferencia entre expectativa inicial y estado real de cierre. Elaboración propia.

Apéndice N: Resumen de artefactos activos del cierre

A continuación se presentan los artefactos activos resultantes del cierre (véase la Tabla 33).

Tabla 33

Resumen de artefactos activos del cierre documental

Artefacto	Ubicación principal	Estado	Uso en defensa o cierre
Informe final actual	Informe_final/informe_final.pdf	Activo	Documento principal para evaluación académica.
Manual de usuario	Informe_final/Manual_de_usuario/manual_usuario.pdf	Activo	Soporte al flujo visible y a la experiencia de uso final.
Manual técnico	Informe_final/Manual_tecnico/manual_tecnico.pdf	Activo	Soporte de arquitectura, módulos y tooling.
Landing y documentación pública	docs/index.html, docs/manuales/	Activo	Punto de acceso a la build, manuales, guías técnicas e instrumentos de evaluación.
Validación agregada e instrumentos	Informe_final/validacion/, docs/manuals/	Activo	Evidencia pública anonimizada y protocolos de evaluación; los registros individuales se conservan por canal académico controlado.
Telemetría técnica agregada	Telemetry/Mediciones_WebGL/	Activo	Exportaciones de profiler y mediciones WebGL usadas para las tablas de rendimiento.
Diagramas Mermaid exportados	Informe_final/figures/chapter4/	Activo	Explican arquitectura, flujo, restricciones y despliegue.
Anexos técnicos seleccionados	Informe_final/Desarrollo_App/Documentacion_Tecnica/	Activo	Documentan arquitectura, despliegue, WebGL y sistema térmico en la forma citada por este informe.

Nota. Esta tabla resume únicamente artefactos activos para consulta pública del cierre; auditorías históricas, bitácoras de trabajo, borradores y registros individuales se conservan como evidencia interna o por canal académico controlado, pero no como fuente primaria pública. Elaboración propia.