Universidad San Jorge

Escuela de Arquitectura y Tecnología

Grado en Ingeniería Informática

Proyecto Final

Análisis y Desarrollo de un plugin en Unity para la carga y modificación de modelos de datos de construcción en tiempo real

Autor del proyecto: Mario González Romero

Director del proyecto: Manuel Ballarín Naya

Zaragoza, 20 de junio de 2025

Este trabajo constituye parte de mi candidatura para la obtención del título de Graduado en Ingeniería Informática por la Universidad San Jorge y no ha sido entregado previamente (o simultáneamente) para la obtención de cualquier otro título.

Este documento es el resultado de mi propio trabajo, excepto donde de otra manera esté indicado y referido.

Doy mi consentimiento para que se archive este trabajo en la biblioteca universitaria de Universidad San Jorge, donde se puede facilitar su consulta.

Firma

Fecha

20 de junio de 2025

Dedicatoria y Agradecimiento

Quiero agradecer, en primer lugar, a todo el profesorado que me ha acompañado durante mi formación, desde la educación básica hasta la universitaria. Todos y cada uno de vosotros habéis puesto vuestro granito de arena para llegar a este momento de la mejor manera posible. Mención especial para mi tutora, África Domingo, por inspirarme a dar siempre el 100 % de mí mismo en los momentos más duros y por ser un apoyo incondicional. Gracias de corazón.

También quiero expresar mi gratitud a Manuel Ballarín por su valiosa orientación en este exigente trayecto, por dedicarme generosamente su tiempo y enseñarme su profesionalidad.

A mis padres, Miguel y Maite, por ser los pilares de mi vida y con vuestra fortaleza brindarme las oportunidades que me han permitido alcanzar este momento. A toda mi familia, porque tengo la mejor familia, tanto por los que hoy me acompañan como por los que ya no están presentes, gracias por vuestro cariño y apoyo incondicional; sin vosotros no sería quien soy. Yayo, ¡al fin lo hemos conseguido!

A mi novia, María, por creer en mí en cada paso del camino, impulsarme a dar siempre mi mejor versión y brindarme tu apoyo incondicional; eres la mejor compañera de vida que uno puede tener.

No puedo olvidar a mis amigos y compañeros, en especial a Rubén, Daniel y Carlos, por sacarme una sonrisa en los momentos más difíciles y motivarnos mutuamente para seguir adelante.

Gracias a todos por estar. Este trabajo también es vuestro.

Tabla de contenido

Resumen

En un contexto donde la eficiencia energética, sostenibilidad y optimización constructiva se imponen, BIM (*Building Information Modeling*) se revela esencial en la transformación digital de la arquitectura, ingeniería y construcción. BIM facilita **representaciones 3D colaborativas** y gestiona activos desde el diseño hasta el mantenimiento, pero su adopción encuentra retos tecnológicos, económicos e interoperabilidad que limitan su integración en entornos inmersivos.

Este proyecto desarrolla un plugin para Unity 6 que carga, visualiza y edita en tiempo real modelos IFC (*Industry Foundation Classes*), apoyando gemelos digitales vía IoT. La solución supera las **restricciones de herramientas comerciales** (altos costes, formatos propietarios y edición limitada) y la dependencia de productos y ecosistemas como Autodesk.

Se describe el proceso de diseño, implementación e integración multiplataforma, fundamentando las decisiones tecnológicas. El resultado es una **plataforma flexible y escalable** para gestionar modelos BIM sin recurrir a licencias costosas, impulsando su adopción en proyectos de construcción e infraestructuras. Desarrollado durante mi labor en Hiberus Tecnología, el plugin ya se aplica en proyectos reales de innovación, orientados a la digitalización avanzada mediante entornos inmersivos.

Palabras clave

Building Information Modeling (BIM), Industry Foundation Classes (IFC), Unity 3D, interoperabilidad, gestión de infraestructuras.

Abstract

In a context where energy efficiency, sustainability, and construction optimization are paramount, BIM (*Building Information Modeling*) proves essential to the digital transformation of architecture, engineering, and construction. BIM enables **collaborative 3D representations** and manages assets from design through maintenance, yet its adoption faces technological, economic, and interoperability challenges that hinder its integration into immersive environments.

This project develops a Unity 6 plugin that loads, visualizes, and edits IFC (*Industry Foundation Classes*) models in real time, supporting digital twins via IoT. The solution overcomes the **limitations of commercial tools** (high costs, proprietary formats, and limited real-time editing) and avoids dependency on vendor ecosystems such as Autodesk.

The design, implementation, and cross-platform integration process is detailed, with all technological decisions justified. The result is a **flexible, scalable platform** for managing BIM models without expensive licenses, driving adoption in construction and infrastructure projects. Developed during my tenure at Hiberus Tecnología, the plugin is already being applied in real-world innovation projects focused on advanced digitalization through immersive environments.

Keywords

Building Information Modeling (BIM), Industry Foundation Classes (IFC), Unity 3D, interoperability, infrastructure management.

1. Introducción

La industria de la construcción atraviesa una profunda **transformación digital**, impulsada por la adopción de metodologías avanzadas como el Building Information Modeling (BIM). Lejos de ser únicamente una herramienta tecnológica, **BIM constituye una metodología colaborativa de trabajo** que permite crear, gestionar y compartir representaciones digitales inteligentes de activos físicos a lo largo de todo su ciclo de vida. Cuando hablamos de activos físicos nos referimos a infraestructuras lineales, edificaciones e instalaciones técnicas; por ejemplo, un puente carretero, un edificio residencial o un sistema de climatización industrial. Estas representaciones integran no solo la geometría tridimensional, sino también **información crítica sobre materiales, costes, tiempos, sostenibilidad y mantenimiento** [1]. Su impacto es tal que líderes mundiales del sector como Sacyr, FCC, Acciona o Ferrovial han adoptado BIM como eje central en sus estrategias de digitalización, aplicándolo de forma transversal en proyectos de edificación, obra civil y mantenimiento de infraestructuras críticas. Esta adopción masiva no solo responde a las exigencias normativas, sino también a los **beneficios tangibles que BIM aporta en términos de eficiencia, trazabilidad y control integral de los proyectos**.

La implantación de **BIM ha demostrado beneficios ampliamente documentados** en términos de reducción de errores, mejora de la coordinación interdisciplinar, optimización de los plazos y disminución de costes [2], [4]. En consecuencia, **numerosos países han establecido marcos regulatorios que exigen su uso en proyectos públicos**. Por ejemplo, el Reino Unido impuso su obligatoriedad en 2016, y España lo hizo en el ámbito de la edificación desde 2018 y en infraestructuras desde 2019 [5]. Al mismo tiempo, organismos internacionales como **buildingSMART promueven estándares abiertos como IFC** (Industry Foundation Classes), que buscan asegurar la interoperabilidad entre distintas plataformas y herramientas del ecosistema BIM.

No obstante**, la integración práctica de BIM en flujos de trabajo digitales modernos sigue enfrentando importantes desafíos**. La implantación efectiva requiere cambios organizativos profundos, inversión en formación especializada y acceso a herramientas que, en muchos casos, están condicionadas por licencias comerciales restrictivas y formatos cerrados [1].

En paralelo**, la irrupción de tecnologías como los motores gráficos en tiempo real** (como Unity o Unreal Engine) **está redefiniendo las posibilidades** de visualización, simulación y edición de modelos BIM. Estas plataformas permiten crear experiencias inmersivas para recorridos virtuales, simulaciones constructivas 4D, mantenimiento predictivo o interacción directa con el modelo por parte de profesionales no técnicos. Sin embargo, **los softwares BIM comerciales (Revit, Archicad, etc.) no ofrecen integración nativa con estos motores**,

lo que obliga a recurrir a exportaciones intermedias (como OBJ), con la consiguiente pérdida de datos, dificultades de sincronización y ausencia de edición en tiempo real.

En este contexto, el presente proyecto propone el **desarrollo de una solución nativa que permita la carga eficiente de modelos BIM en Unity 6, preservando tanto su geometría como sus metadatos**, y habilitando la modificación directa en tiempo real de los metadatos principales como el identificador IFC, el tipo de material, las dimensiones, la fase de construcción y la información de costes. Para ello, **se ha implementado un plugin propio basado en librerías BIM de código abierto**, como xBIM Essentials e IfcOpenShell, con el objetivo de garantizar independencia tecnológica, interoperabilidad y reducción de costes.

La solución propuesta facilitará la gestión de modelos BIM directamente en obra, permitiendo a los agentes implicados revisar, modificar o enriquecer la información de manera interactiva y colaborativa, sin necesidad de recurrir a herramientas privativas. Además, abrirá la puerta al desarrollo de aplicaciones personalizadas (como simuladores de mantenimiento o presentaciones interactivas para clientes) sin depender de terceros para su implementación.

En última instancia, y de la mano de Hiberus, este proyecto busca contribuir a la **democratización del uso de BIM** en entornos inmersivos y a la consolidación de flujos de trabajo digitales más abiertos, sostenibles y adaptados a las necesidades reales del sector de la construcción. Se añade una ilustración descriptiva del proyecto general con el fin de facilitar su entendimiento.

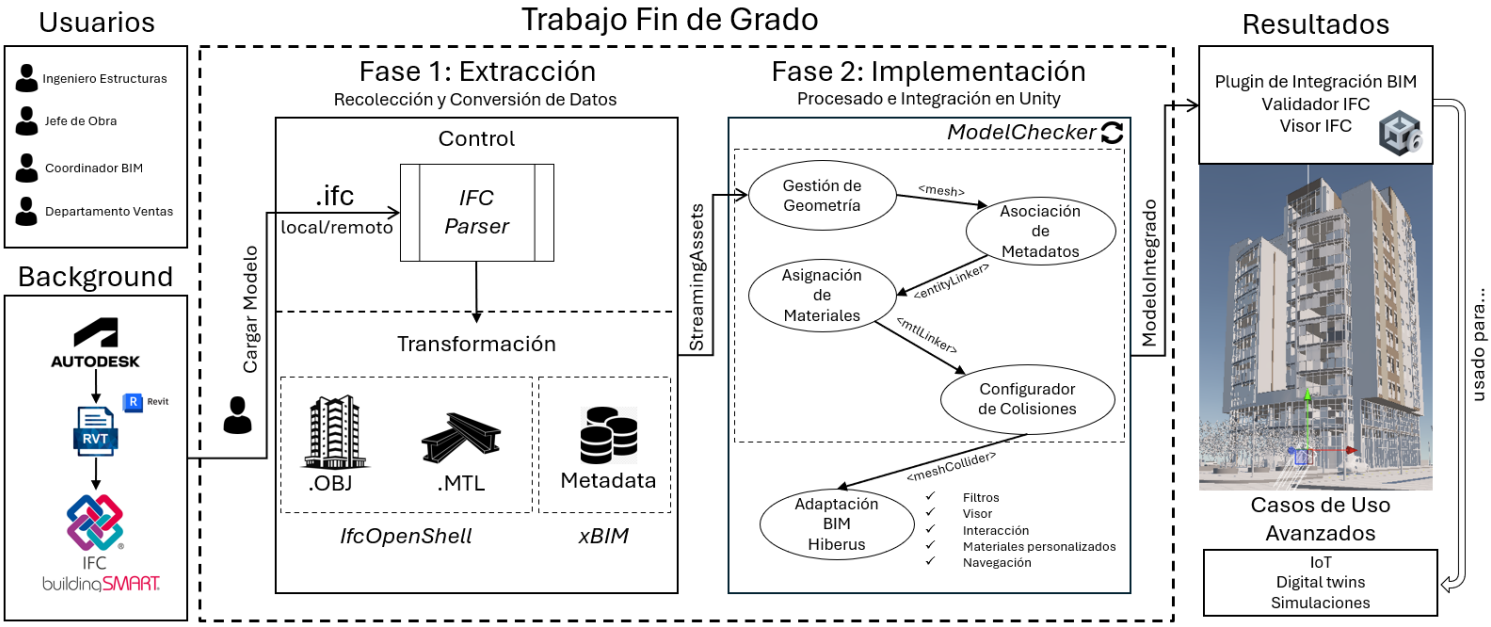

Ilustración 1: "Figura general descriptiva de la solución".

1.1. Contexto general y motivación

El entorno actual en la construcción se caracteriza por la creciente demanda de eficiencia, sostenibilidad y precisión en los procesos, como la planificación 4D, la gestión de recursos y el mantenimiento predictivo. La digitalización mediante BIM se ha impuesto como una **herramienta esencial que mejora la coordinación entre disciplinas y reduce los errores** comunes en los métodos tradicionales en 2D [1]. La adopción de BIM, por parte de las empresas del sector, por ejemplo, Ferrovial o Acciona; entre otras, **se ha acelerado en muchos países gracias a normativas y mandatos institucionales**; por ejemplo, en el Reino Unido se ha exigido el uso de BIM Nivel 2 en proyectos públicos desde 2016, y en España la obligatoriedad en licitaciones públicas comenzó en 2018, extendiéndose obligatoriamente a infraestructuras a partir de 2019 [5].

Esta evolución normativa evidencia la **relevancia de BIM como estándar para garantizar eficiencia y transparencia en la ejecución de proyectos**. Al mismo tiempo, la digitalización abre nuevas oportunidades, entre ellas la integración de BIM con tecnologías emergentes como la *realidad virtual* y los *motores de videojuegos*. Herramientas como Unity permiten **desarrollar entornos 3D interactivos en tiempo real**, lo que posibilita la visualización y edición directa de modelos BIM, ofreciendo experiencias innovadoras para el seguimiento de obras y la toma de decisiones colaborativas [1], [2]. Esta convergencia tecnológica es especialmente **atractiva para empresas que buscan optimizar sus procesos y reducir costes**, al mismo tiempo que ofrecen a sus clientes y equipos de trabajo herramientas de última generación para la gestión de proyectos.

1.2. Planteamiento del problema

Aunque la metodología Building Information Modeling (BIM) ha demostrado beneficios significativos en la gestión integral de la información a lo largo del ciclo de vida de proyectos constructivos, **su integración con plataformas de visualización en tiempo real sigue enfrentando barreras técnicas relevantes**. En particular, las herramientas comerciales más utilizadas, como Autodesk Revit o Graphisoft ArchiCAD, no ofrecen compatibilidad nativa con motores gráficos interactivos como Unity o Unreal Engine, lo que obliga a realizar exportaciones a formatos intermedios (como FBX u OBJ) que, en muchos casos, suponen la pérdida de metadatos y propiedades del modelo [1], [2].

Adicionalmente, cualquier modificación realizada en el entorno interactivo (por ejemplo, durante una revisión en obra o una simulación) debe ser posteriormente replicada en el software BIM original, interrumpiendo los flujos de trabajo y dificultando la colaboración entre perfiles técnicos y no técnicos [2], [4]. Esta **falta de interoperabilidad dinámica crea un desfase crítico**

entre la capacidad de visualizar modelos BIM en tiempo real y la posibilidad de modificarlos de forma interactiva y contextual, lo que representa una limitación importante en entornos de construcción modernos, donde la inmediatez y la adaptabilidad son factores clave.

Como consecuencia, **existe una necesidad tecnológica no resuelta en el sector: habilitar la carga y edición directa de modelos BIM en motores 3D en tiempo real**, sin depender de herramientas comerciales restrictivas ni procesos intermedios que comprometan la integridad de la información. Esta problemática es especialmente relevante para empresas que buscan adoptar flujos de trabajo ágiles y escalables mediante tecnologías de código abierto o personalizadas.

En este contexto, la cuestión central que motiva el presente Trabajo Fin de Grado es:

¿Cómo puede integrarse un modelo BIM en un motor gráfico en tiempo real, permitiendo su modificación interactiva, sin depender de soluciones comerciales cerradas y costosas?

La resolución de este problema implica el diseño y desarrollo de un plugin nativo para un motor 3D, capaz de importar modelos BIM completos (incluyendo geometría, materiales y metadatos) y permitir su edición en tiempo real dentro del entorno interactivo. Esta solución no solo permitiría una mayor flexibilidad en el uso de BIM en obra, sino que facilitaría nuevas aplicaciones colaborativas, como simuladores de mantenimiento, entornos formativos o visualizaciones orientadas al cliente, manteniendo siempre la trazabilidad y fidelidad del modelo original [1], [5].

1.3. Organización del documento

El documento se divide en once capítulos que acompañan al lector desde el planteamiento inicial hasta la documentación complementaria. El primero introduce la motivación, el problema y la organización de los contenidos; el segundo revisa el estado del arte en metodología BIM, herramientas disponibles y justifica la elección de librerías y motor gráfico; el tercero define los objetivos general y específicos; el cuarto profundiza en el formato IFC, tanto en su estructura alfanumérica como en las representaciones geométricas STEP; el quinto describe la metodología de trabajo y las herramientas de apoyo; el sexto detalla el desarrollo en siete sprints con sus tareas, resultados y ajustes; el séptimo realiza un estudio económico con estimación de costes, comparación con soluciones comerciales y evaluación de beneficios; el octavo presenta los resultados del plugin, su validación funcional, rendimiento y comparación con herramientas comerciales; el noveno expone las conclusiones, las limitaciones y posibles mejoras; el décimo recopila la bibliografía utilizada; y el undécimo incluye los anexos (propuesta de proyecto, actas, índices de ilustraciones y tablas, glosario).

2. Antecedentes y Estado del Arte

Esta sección presenta un **análisis de la evolución y adopción** de BIM en la industria de la construcción, enfatizando tanto los beneficios como las limitaciones de las herramientas comerciales existentes. Se abordarán los desafíos asociados con el uso de software propietario, como altos costes, restricciones de licenciamiento y problemas de interoperabilidad, y se explorarán las alternativas basadas en estándares abiertos (por ejemplo, el más utilizado como norma general, el *formato IFC*) y soluciones *open source*. Este análisis permite identificar la necesidad de desarrollar herramientas personalizadas y flexibles, que integren entornos inmersivos y que sean capaces de *superar las limitaciones actuales*, sentando así las bases teóricas para el desarrollo del plugin propuesto.

2.1. Building Information Modeling (BIM) y su relevancia actual

El término *Building Information Modeling (BIM)* hace referencia tanto al proceso colaborativo de generación y gestión de información de una construcción, como al modelo digital resultante que centraliza dicha información [1]. En esencia, **BIM es una evolución del CAD hacia modelos inteligentes**.

En lugar de planos independientes, se construye un **modelo tridimensional paramétrico que integra geometría y datos alfanuméricos** de los elementos constructivos (muros, vigas, instalaciones, etc.). Cada elemento del modelo BIM contiene información rica (propiedades físicas, costos, fase de obra, fabricante, mantenimiento, etc.), por lo que el **modelo se convierte en una base de conocimiento compartida para todos los actores del proyecto**.

Una característica clave es que **el modelo es único**; cualquier cambio en un elemento se refleja automáticamente en todas las vistas y planos derivados, **manteniendo la consistencia documental**. Esto contrasta con los métodos tradicionales, donde la incoherencia entre planos podía provocar errores. En resumen, **BIM abarca el ciclo de vida completo de un activo construido**, desde el diseño conceptual y la planificación, pasando por la construcción, hasta la operación y mantenimiento del edificio o infraestructura [1]. Su alcance trasciende el 3D: se suele hablar de *4D* (3D + tiempo), *5D* (coste), *6D* (sostenibilidad), e incluso *7D* (gestión) asociados al modelo BIM como se puede apreciar en la *Ilustración 2*. Además de las siete dimensiones estándar ahora en el mundo BIM hay un debate abierto que implica 8D (seguridad), 9D (no pérdidas) y 10D (industrialización).

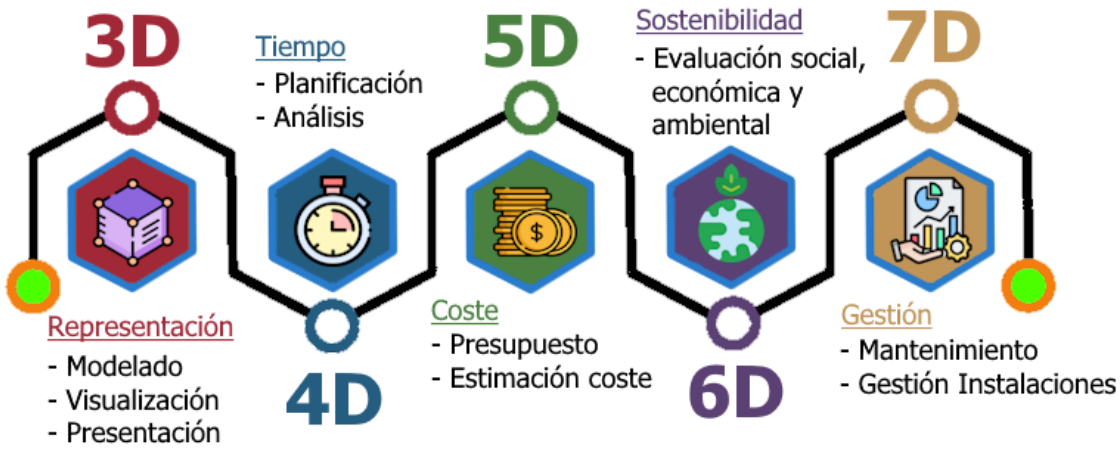

Ilustración 2: "De 3D a 7D: Las Dimensiones Clave en BIM".

2.1.1. Beneficios de la implementación de BIM

La adopción de BIM aporta numerosos beneficios documentados en la literatura y la práctica. En primer lugar, **mejora la coordinación interdisciplinar** y la comunicación dentro del equipo de proyecto. Al trabajar todos sobre el mismo modelo digital, **se reduce la duplicidad de datos y los errores por falta de información actualizada**. Por ejemplo, arquitectos, estructurales e ingenieros de instalaciones coordinan sus diseños en un entorno común, detectando interferencias (por ejemplo, una viga que colisiona con una tubería) antes de llegar a obra. Estudios reportan que **la detección de conflictos mediante BIM puede ahorrar hasta un 10% del valor del contrato en costes evitados** [25]. En segundo lugar, **BIM agiliza los procesos**: tareas como generar planos, mediciones y presupuestos se automatizan extrayendo la información del modelo, lo que conlleva procesos más rápidos y efectivos [1]. Se ha medido, por ejemplo, que preparar un presupuesto con BIM puede ser hasta un 80% más rápido que con métodos tradicionales, y con una precisión de cálculo de materiales muy alta (desviaciones menores al 3% respecto a estimaciones finales) [1].

En tercer lugar, BIM **permite mejorar la calidad del diseño**: al poder analizar el edificio virtualmente (simulaciones estructurales, energéticas, recorridos 3D, etc.), los proyectistas pueden optimizar soluciones y reducir cambios durante la construcción [1]. Esto redunda en menos órdenes de cambio y menos trabajos no previstos; se estima que BIM puede eliminar hasta el 40% de las modificaciones no planificadas en obra [1]. Otros beneficios destacados en proyectos BIM son la **reducción de residuos y retrabajos** (al haber menos errores), la mejora en la seguridad (mediante secuenciación 4D para planificar la obra) y un mejor feedback para el propietario, que recibe al final un modelo "*as-built*" rico en información para la gestión del activo

reduciendo tiempos extra. En suma, BIM contribuye a proyectos más económicos, rápidos y fiables, cumpliendo la tan buscada "*triple restricción* o **triángulo de hierro** (modelo desarrollado por Martín Barnes en 1969)" de coste, tiempo y calidad en construcción. Bryde et al. analizaron 10 proyectos y encontraron impactos positivos de BIM en todos ellos, especialmente en la reducción de costes y el control de plazos [4]. Asimismo, Azhar resume que BIM produce procesos más eficientes, mejores diseños y mayor productividad, entre otros beneficios clave [1]. Cabe mencionar que la magnitud de estos beneficios puede variar según el grado de implementación BIM en el proyecto (nivel de madurez BIM) y la competencia del equipo en su uso.

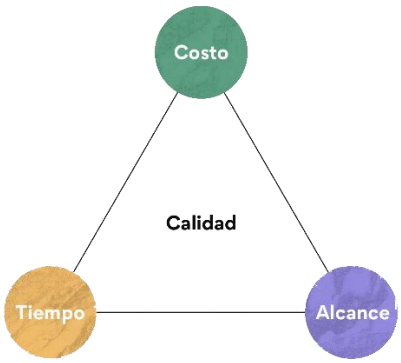

Ilustración 3: "Triángulo de hierro". Idea desarrollada por Martín Barnes en 1969. Propiedad de Imagen: asana.com

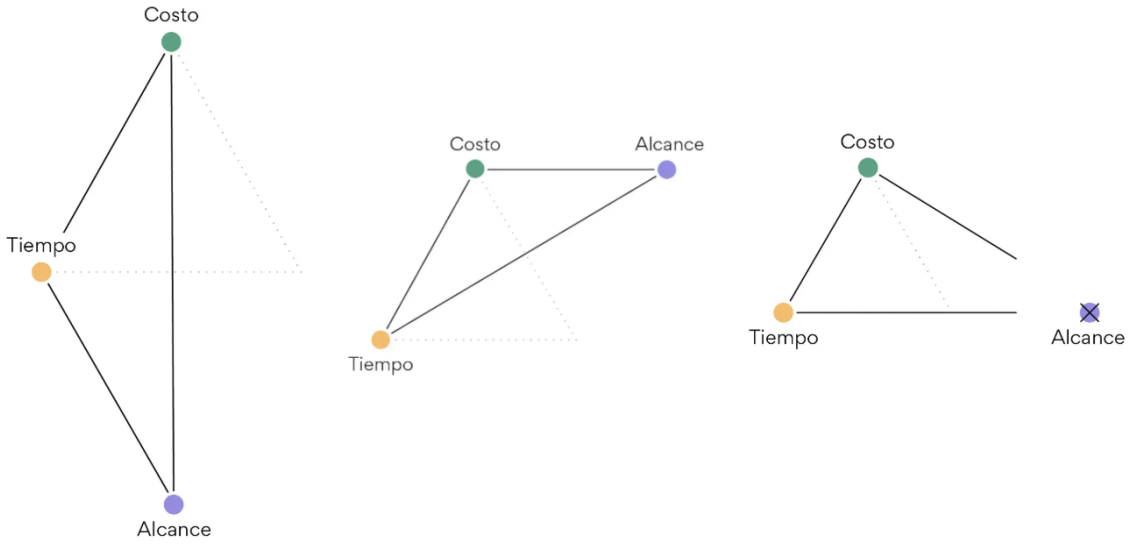

Ilustración 4: "No puedes aumentar el alcance sin aumentar también el costo o el tiempo". Idea de Bornes (1969). Elaboración propia basada en la idea de Bornes.

El objetivo en todo proyecto debe ser una **balanza perfectamente equilibrada** como puede visualizarse en la *Ilustración 5* entre los activos: costo, tiempo y alcance; por lo que es muy

importante definir previamente los puntos correctamente a la hora de diseñar y planificar un desarrollo aplicado a la gestión de proyectos.

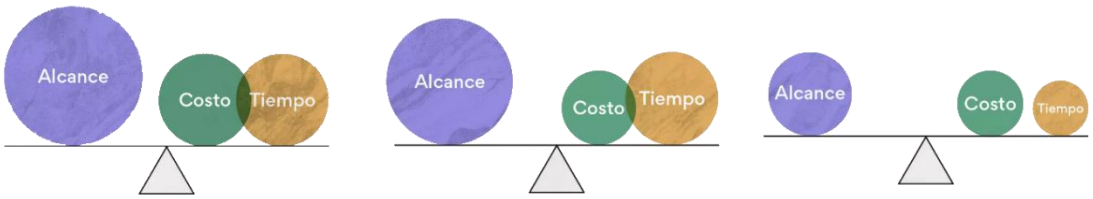

Ilustración 5: "El equilibrio objetivo". Muestra el equilibrio objetivo de la triple restricción que hay que acordar, valorar y alcanzar en cualquier desarrollo BIM. Elaboración propia.

2.1.2. Relevancia actual de BIM

En la última década, **BIM ha pasado de ser una tecnología emergente para convertirse en estándar en la industria AEC** (Architecture, Engineering & Construction). Muchos países han desarrollado estrategias nacionales para impulsar BIM, reconociendo sus ventajas en eficiencia. En Europa, la Directiva 2014/24/UE fomentó el uso de BIM en proyectos financiados con fondos públicos [5]. Países como el Reino Unido, como se ha indicado anteriormente, fueron pioneros: desde 2016 se exige BIM Nivel 2 en todos los encargos públicos, lo que ha posicionado al Reino Unido como líder mundial en adopción de BIM. **En España, se creó en 2015 la Comisión esBIM para promover la incorporación de BIM**, y la Ley 9/2017 de Contratos del Sector Público abrió la puerta a exigir BIM en licitaciones públicas [5]. Como resultado, desde diciembre de 2018 es obligatorio presentar modelos BIM en ciertos proyectos públicos de edificación, y desde julio de 2019 en obras de infraestructuras [5].

Estas medidas han impulsado a empresas y administraciones a capacitarse en BIM. A nivel global, Estados Unidos, Canadá, Australia y China también muestran altos índices de adopción de BIM [1], mientras que regiones como Oriente Medio y Latinoamérica avanzan de forma más gradual. En cualquier caso, la tendencia es clara: **BIM se está convirtiendo en la piedra angular para la digitalización de la construcción**. Su relevancia actual no solo se refleja en la fase de diseño, sino en todo el ciclo de vida: las empresas constructoras emplean BIM para planificar la obra (simulando procesos constructivos), *los* gestores de activos lo utilizan para mantenimiento (Facility Management), e incluso se enlaza BIM con sistemas IoT para crear gemelos digitales (*digital twins*) de edificios en operación [22]. Un ejemplo destacado es el Plan BIM 2020 de infraestructuras de Alemania, que ha establecido BIM como requisito en proyectos de infraestructura a partir de ese año [5].

BIM ha pasado de ser una innovación tecnológica para consolidarse como una práctica habitual y, en muchos casos, obligatoria en proyectos contemporáneos, gracias a las importantes mejoras que aporta en productividad, reducción de riesgos y sostenibilidad en la industria de la construcción. No obstante**, la evolución de BIM continúa avanzando hacia una mayor integración tecnológica**, adoptando filosofías como Lean Construction, el uso de Digital Twins, IoT, Big Data e incluso y la que consideran más importante la Inteligencia Artificial, lo que promete revolucionar aún más la gestión y operación de proyectos constructivos en los próximos años.

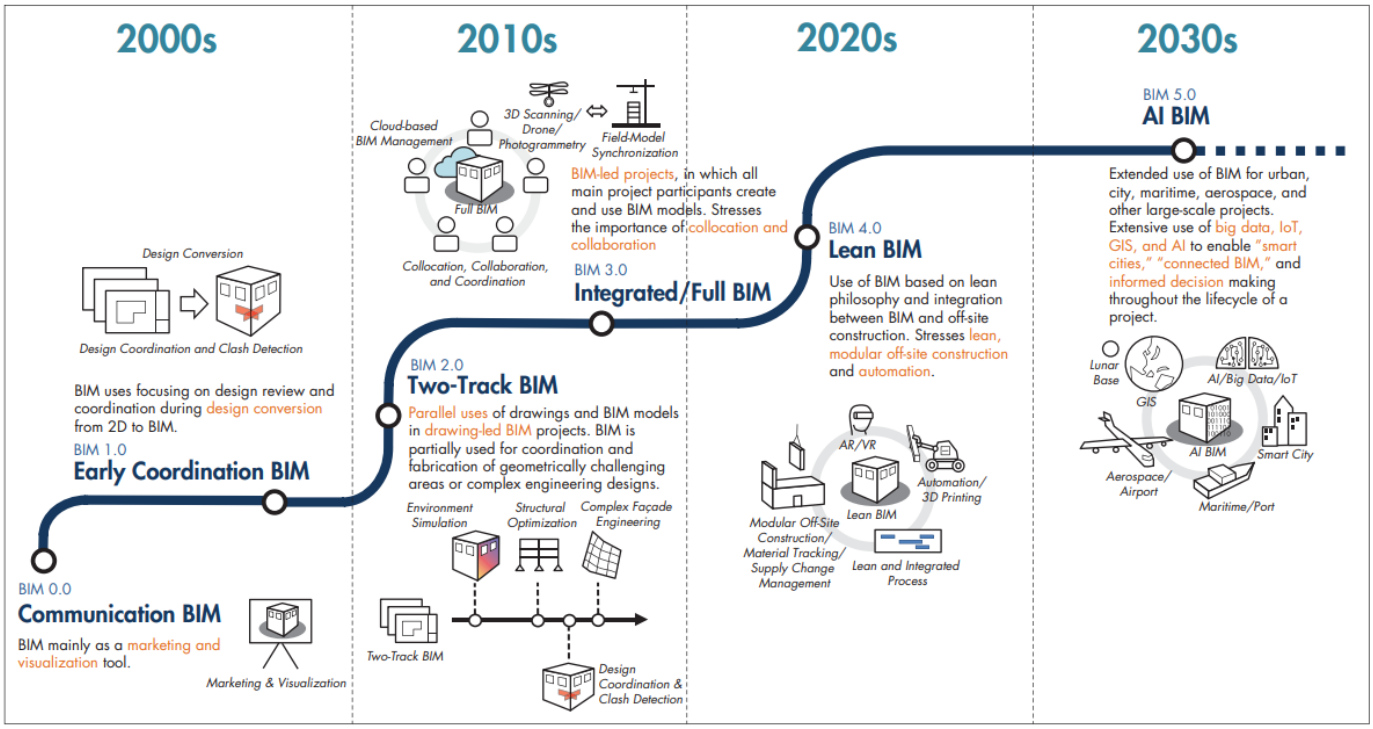

Ilustración 6: "Progresión de BIM".

2.1.3. Caso de estudio: Grupo LOBE

Grupo LOBE es una promotora-constructora española con sede en Zaragoza, dedicada a la edificación residencial, que desde 2009 ha emprendido una profunda transformación de sus procesos mediante la **adopción integral de la metodología BIM en todos sus proyectos**. La empresa identificó desde los inicios los beneficios de la visualización tridimensional y la coordinación que ofrece BIM, pero fue más allá al explotar al máximo la "I" de Información, incorporando datos precisos en sus procesos productivos para mejorar la estandarización y el control de sus desarrollos. No obstante, pronto detectaron que las herramientas comerciales disponibles no satisfacían todas sus necesidades internas, dado que requerían integrar el modelo BIM con sus propios sistemas de planificación y control de producción, lo que motivó la búsqueda de soluciones a medida [6].

Ante estas limitaciones, Grupo LOBE decidió desarrollar una plataforma BIM personalizada denominada **HUBE**, construida sobre Autodesk Navisworks y utilizando como base los modelos exportados desde Revit. Esta plataforma creó un entorno unificado en el que arquitectos, ingenieros y jefes de obra pueden compartir y actualizar el modelo BIM en todas las fases del proceso constructivo, garantizando la coherencia y la disponibilidad de la información para todos los agentes implicados [6].

La solución interna **HUBE incorpora funcionalidades a medida que potencian la eficiencia operativa** de Grupo LOBE, tales como el control de fabricación, la asignación automatizada de tareas y el seguimiento económico en tiempo real, todo ello vinculado al modelo BIM central. Gracias a esta integración, la compañía ha impulsado, junto con la filosofía Lean, una profunda reestructuración de sus procesos de gestión, añadiendo valor y optimizando el uso de recursos en sus proyectos inmobiliarios, lo que se ha traducido en una mayor calidad, agilidad y reducción de costes [6].

Este caso de éxito pone de manifiesto varios aspectos relevantes para el sector de la construcción: en primer lugar, la importancia estratégica de BIM como factor de competitividad (permitiendo a Grupo LOBE ofrecer viviendas de mayor calidad en plazos reducidos); en segundo lugar, las carencias de las herramientas comerciales que pueden conducir al desarrollo de soluciones propias; y, finalmente, el valor de combinar BIM con enfoques complementarios como Lean Construction e IPD para lograr mejoras integrales en productividad. Asimismo, la experiencia de LOBE ejemplifica cómo la inversión en talento y competencias tecnológicas, con la creación de perfiles profesionales especializados (por ejemplo, modeladores BIM integrados en obra), permite adelantarse a la curva de adopción de BIM y consolidarse como referente nacional. En síntesis, su trayectoria demuestra que BIM, cuando se aprovecha plenamente, puede actuar como un auténtico **catalizador de innovación**, aunque a veces requiera integrar y personalizar herramientas más allá de las ofertas del mercado [6].

2.2. Herramientas existentes para la gestión de modelos BIM

En el panorama actual **existen numerosas herramientas software para crear, gestionar y visualizar modelos BIM**. En la etapa de diseño, los programas más difundidos son Autodesk Revit, Graphisoft ArchiCAD y Bentley OpenBuildings (antes AECOsim), que permiten modelar edificios en 3D con información paramétrica. Estas plataformas de authoring BIM dominan el mercado y suelen ser la fuente de los modelos digitales. Otras herramientas destacadas incluyen Tekla Structures (orientada a estructuras e ingeniería civil), Allplan de Nemetschek, Vectorworks y soluciones específicas por disciplina (por ejemplo, Civil 3D o InfraWorks para infraestructuras). Junto a los programas de creación, **hay aplicaciones especializadas en el análisis y**

coordinación de modelos BIM: Autodesk Navisworks para revisión interdisciplinar y detección de choques, Solibri Model Checker para validación de normas de calidad, Synchro PRO para simulaciones 4D de planificación, entre otras. También **existen visores BIM independientes que permiten explorar modelos IFC sin tener el software de autor**: por ejemplo, Solibri Anywhere, Trimble Connect, BIMcollab Zoom o BIM Vision.

Estas herramientas facilitan compartir el modelo con clientes o contratistas que no poseen licencias de los programas de autoría. Por último, han emergido plataformas colaborativas en la nube, como Autodesk BIM 360 / ACC o Bentley ProjectWise, que gestionan la CDE (Common Data Environment) del proyecto, permitiendo trabajo multiusuario sobre modelos BIM, control de versiones y comentarios en línea; para más información acerca del software para BIM vea la *Tabla 1*. En conjunto, el ecosistema de software BIM es amplio y heterogéneo, cubriendo desde el diseño hasta la construcción y operación. Sin embargo, cada aplicación pertenece generalmente a un fabricante y maneja su propio formato de datos (por ejemplo, Revit usa RVT, ArchiCAD usa PLN). Para intercambiar modelos entre distintas herramientas se utiliza el formato estándar *IFC (Industry Foundation Classes)*, un formato abierto desarrollado por buildingSMART. El IFC actúa como puente neutral para lograr cierta interoperabilidad entre plataformas; aun así, en la práctica la integración total de datos no es trivial y a menudo se requieren flujos de trabajo específicos o complementos adicionales para transferir toda la información de un software BIM a otro.

Software BIM	Puntos de aplicación	Formatos compatibles	Costo estimado
Autodesk Revit	Diseño arquitectónico, planificación de construcción y colaboración en visualización	.ifc, .rvt, .dwg, .fbx, .dgn	3.338€/año/usuario
Rhino	Producción de animación tridimensional	.ifc, .dwg, .dxf, .3ds, .skp, .lwo, .obj	995€/usuario
Autodesk 3ds Max	Renderizado de animación 3D	.ifc, .3ds, .dwg, .ai, .xml, .fbx	2.263€/año/usuario

Autodesk Navisworks Manage	Simulación, gestión de construcción, visor	.ifc, .nwc, .rvt, .dwg, .dxf, .dgn, .nwd	3.195€/año/usuario
Autodesk Civil 3D	Modelado de terrenos, gestión de datos	.ifc, .fgbh, .ras, .xml, .txt	3.237€/año/usuario
Tekla Structures	Diseño detallado de estructuras de acero y simulaciones de construcción	.ifc, .dwg, .dxf, .skf, .dgn, .std	3.048€/año/usuario
Unity Reflect (Industry)	Visualización de modelos BIM en Unity	.ifc, .rvt, .fbx, .nwc	4.554€/año/usuario
BIMMAKE	Modelado de construcción, diseño detallado	.ifc, .skp, .pdf, .rvt	No disponible públicamente

Tabla 1: Comparación de software BIM y sus costos. Elaboración propia con información de los sitios web oficiales.

Se observa como indica la anterior tabla que la mayoría de **las soluciones presentan modelos de licenciamiento costosos**, lo que motiva la exploración de alternativas de código abierto para la gestión y visualización de modelos IFC.

2.3. Limitaciones de las herramientas comerciales

A pesar de la madurez y coste de las herramientas BIM comerciales líderes, presentan una serie de **limitaciones importantes que motivan la búsqueda de soluciones alternativas más abiertas y flexibles**, especialmente para aplicaciones especializadas como la de este proyecto. A continuación, se resumen las principales limitaciones identificadas:

2.3.1. Costos elevados y modelos de licenciamiento restrictivos

El software BIM profesional suele implicar licencias de costo muy alto (vea *sección 2.2*) y esquemas de suscripción poco accesibles para estudiantes, PYMEs o desarrolladores independientes. Estudios sobre adopción BIM señalan que el **alto costo del software y la capacitación es uno de los obstáculos más significativos para muchas empresas** [1]. Por ejemplo, en una encuesta en Oriente Medio, el factor económico (licencias y hardware) fue

citado como la barrera principal para implementar BIM [20]. Además, los fabricantes imponen condiciones restrictivas de uso: **las licencias suelen ser por puesto o por periodo, limitando la disponibilidad de la herramienta** para un equipo o encareciéndolo todavía más. Esta situación puede frenar la innovación, ya que desarrollar complementos o integraciones sobre software propietario requiere invertir en dichas licencias y ceñirse a las APIs que el proveedor ofrece.

2.3.2. Dependencia de ecosistemas cerrados e interoperabilidad limitada

Muchas **plataformas BIM comerciales funcionan mejor dentro de su propio ecosistema de productos, pero presentan dificultades para comunicarse con herramientas de terceros**. Por ejemplo, Autodesk favorece flujos entre Revit, Navisworks y BIM 360 (plataformas propias de Autodesk), pero la integración con software externo vía IFC no siempre es completa (puede haber pérdida de información o geometrías mal interpretadas en la interoperabilidad). La información creada en un formato propietario no es plenamente reutilizable en otras aplicaciones, generando lock-in que hay que tratar de solventar o disminuir. Esta **dependencia de ecosistemas cerrados preocupa a la industria**, ya que "la falta de interoperabilidad entre diferentes softwares BIM se considera uno de los obstáculos más críticos" para el éxito de BIM [20]. Si los datos no fluyen libremente, se pierden beneficios de eficiencia. Aunque el formato IFC mitiga en parte el problema, no cubre al 100% todas las funcionalidades nativas de cada software, y las implementaciones de IFC varían, **resultando en que un mismo modelo puede verse ligeramente distinto al abrirlo con diferentes programas**. En definitiva, los entornos cerrados comprometen la interoperabilidad, obligando muchas veces a los equipos a adoptar una única marca de herramientas en todo el proyecto. Esto contraviene el principio de open BIM, que busca entornos agnósticos de fabricante.

2.3.3. Falta de flexibilidad y personalización

Las herramientas comerciales BIM están diseñadas para usos generalistas y con interfaces genéricas, por lo que adaptarlas a flujos de trabajo *ad hoc* o integrarlas con sistemas propios puede ser complejo o directamente imposible. La **lógica interna de software como Revit o ArchiCAD es cerrada**; si bien ofrecen APIs para desarrollar complementos, estas interfaces solo permiten cierto grado de extensión. Existen necesidades específicas (por ejemplo, conectar el modelo BIM con una base de datos particular de la empresa o implementar un cálculo a medida sobre los datos) que no pueden resolverse sin acceso al código fuente, lo cual no es posible en software propietario. Un caso ilustrativo fue el de Grupo LOBE: ningún software del mercado ofrecía la integración de procesos que ellos requerían (gestión de compras, planificación Lean vinculada al BIM, etc.), lo que los llevó a crear HUBE para rellenar ese vacío [6]. Asimismo, la

personalización de la interfaz o automatizaciones avanzadas suelen estar limitadas por lo que el proveedor haya previsto. Esta falta de flexibilidad dificulta la innovación y la adaptación de BIM a nichos nuevos. Por el contrario, un entorno abierto permitiría a desarrolladores incorporar funcionalidades nuevas no contempladas originalmente.

2.3.4. Limitaciones en la edición y actualización en tiempo real

Los **programas BIM tradicionales no están concebidos para operar en tiempo real en entornos interactivos** 3D más allá de su propia interfaz. Si bien soportan trabajo multiusuario (por ejemplo, Revit Worksharing o plataformas en la nube que sincronizan cambios), estos **funcionan mediante sincronizaciones periódicas**, no con actualización instantánea continua al estilo de los motores de juego. No existe la posibilidad de, por ejemplo, mover un elemento del edificio en una aplicación externa y que dicho cambio se refleje inmediatamente en el modelo BIM fuente. Cualquier modificación requiere abrir el archivo BIM en el software de autor, editar y exportar de nuevo. Esta rigidez contrasta con las expectativas de interactividad en entornos virtuales modernos. Por ejemplo, Unity permite mover objetos o cambiar parámetros mientras la aplicación corre, viendo el resultado al instante, pero no puede hacerlo con un elemento BIM porque no tiene conexión directa con el modelo original. Existe un claro desfase tecnológico entre el mundo BIM y los motores de videojuegos en este sentido. Investigaciones recientes han explorado vínculos bidireccionales BIM-motor de juego: Mosler y Steitz propusieron un flujo para conectar Unity con Revit en tiempo casi real, exportando cambios de Unity a Revit mediante archivos intermedios [8]. Concluyen que **lograr una sincronización en tiempo real totalmente bidireccional es desafiante por las diferencias de plataformas**, pero resaltan los beneficios potenciales en ahorro de tiempo y colaboración si se consigue superar esa barrera [8]. En resumen, las herramientas actuales no permiten edición interactiva en línea del modelo BIM, lo cual limita aplicaciones como la realidad virtual colaborativa o la revisión de diseño en vivo con múltiples stakeholders.

2.3.5. Futuro incierto de algunas soluciones comerciales utilizadas actualmente

El mercado BIM evoluciona rápidamente, con **fusiones**, **cambios estratégicos** y nuevas plataformas en la nube que ponen en duda la continuidad de herramientas consolidadas. Por ejemplo, Autodesk Navisworks ve cómo su rol podría solaparse con Model Coordination en BIM 360/ACC, y BIM 360 Team finalizó en 2024 en favor de **Autodesk Construction Cloud** [9], obligando a migraciones o dejando sin soporte a sus usuarios. En el caso del software propietario, decisiones del proveedor, subidas de precio, cambios de formato o descontinuación, pueden impactar gravemente en una empresa. En cambio, las soluciones de código abierto suelen perdurar gracias a la comunidad, incluso si el mantenedor original abandona el proyecto. Ante

esta incertidumbre, cobra sentido buscar alternativas **más estables y controlables**; por ello, este proyecto renuncia al software privativo y se orienta a tecnologías abiertas para integrar BIM con Unity, explorando y justificando la elección de librerías open source para manejar datos IFC.

2.4. Factores limitantes para la aplicación de BIM

Además de las limitaciones específicas propias de las herramientas comerciales, **existen factores generales que dificultan la adopción y aplicación efectiva de la metodología BIM** en la industria de la construcción. Estos factores se pueden clasificar en distintas categorías y representan barreras ampliamente reconocidas en la literatura especializada.

La *Ilustración 7*, adaptada del estudio de Sun et al. [3], sintetiza gráficamente estos aspectos y proporciona una visión estructurada sobre los factores limitantes que deben abordarse o mitigarse para facilitar una implementación más efectiva de la metodología BIM.

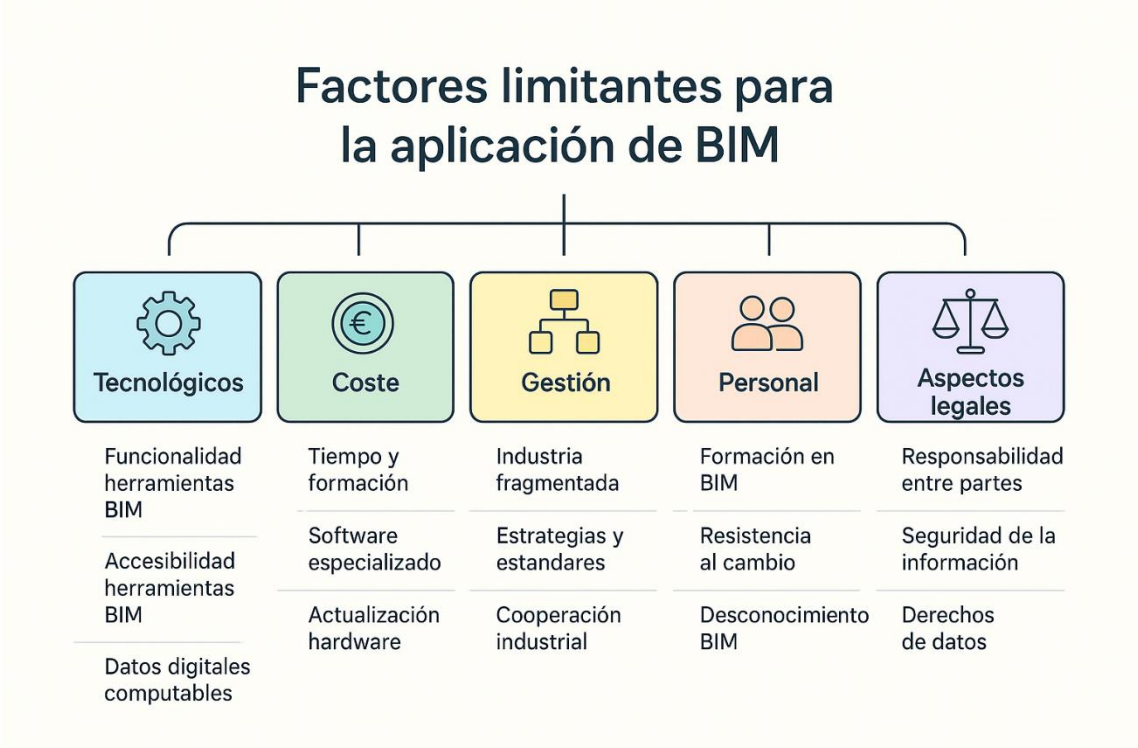

Ilustración 7: "Factores limitantes para la aplicación de BIM". Imagen de elaboración propia basada en [3].

Seguidamente, se describen brevemente las categorías de factores mostradas en la ilustración [3], [20]:

- **Factores tecnológicos** relacionados con la funcionalidad, accesibilidad, gestión y compatibilidad técnica de las herramientas BIM.

- **Factores económicos** debido al elevado coste inicial tanto en formación como en adquisición y actualización de hardware y software específico.
- **Factores de gestión** mostrando dificultades derivadas de la estructura fragmentada del sector, falta de estrategias prácticas y estándares sólidos, así como escasa cooperación entre socios.
- **Factores personales** incluyendo barreras relacionadas con la formación insuficiente de los profesionales, resistencia al cambio, y falta de conocimiento sobre las capacidades de BIM.
- **Factores legales y jurídicos** complejos como la responsabilidad compartida entre las partes, propiedad intelectual sobre los datos BIM, seguridad y fiabilidad de la información, y carencia de protocolos claros.

2.5. Evaluación de alternativas y justificación de la elección

Existen varios proyectos de **código abierto** orientados a trabajar con modelos IFC, el formato abierto estándar de BIM. Entre los más conocidos destacan: BIMserver, una plataforma en Java que actúa como servidor de modelos IFC y permite almacenarlos y consultarlos vía web; IfcOpenShell, una librería en C++/Python para leer, editar y generar ficheros IFC; xBIM Toolkit, un conjunto de librerías .NET para manipular datos BIM; y otros como IfcPlusPlus, OpenIFCTools, etc. [7].

Cada alternativa tiene un enfoque distinto: BIMserver, por ejemplo, está pensado para la gestión centralizada de modelos y colaboración; mientras que IfcOpenShell y xBIM son toolkits orientados a desarrolladores, para ser embebidos en aplicaciones propias. Tras evaluar estas opciones, se decidió enfocar el proyecto en el uso combinado de xBIM e IfcOpenShell, por las siguientes razones principales:

- Unity utiliza C# como lenguaje de scripting, por lo que con una librería nativa .NET como xBIM existe una **compatibilidad natural**.
- xBIM ofrece una API robusta para acceder y manipular la estructura de datos IFC (entidades, propiedades, relaciones), mientras que IfcOpenShell provee un motor geométrico potente basado en Open Cascade para generar la geometría 3D de los elementos a partir de definiciones IFC. Combinando ambas es posible cubrir tanto la **lectura/escritura** de datos BIM como la conversión a geometría renderizable.
- Ambos proyectos son **open source** (licencias LGPL/MIT), **con comunidades activas de desarrolladores**. IfcOpenShell forma parte del ecosistema OSArch y se utiliza en herramientas como BlenderBIM, lo que garantiza continuidad y mejoras constantes;

xBIM, por su parte, es un proyecto maduro, utilizado en aplicaciones industriales y académicas, demostrando alta fiabilidad.

- Al ser bibliotecas nativas (C# y C++), aprovechan mejor el hardware que soluciones interpretadas. xBIM implementa su núcleo geométrico en C++ para operaciones de gran carga computacional [10], y IfcOpenShell está escrito en C++ y expuesto a Python, combinando **rendimiento en cálculo geométrico** con la flexibilidad de un lenguaje de scripting [11]. Esto resulta adecuado para manejar modelos grandes en tiempo real.

Otras alternativas quedaron descartadas por diversos motivos. Por ejemplo, BIMserver es excelente como repositorio IFC multiusuario, pero integrar un servidor Java completo dentro de Unity habría sido complejo y sobredimensionado para nuestro objetivo (se busca carga local en la aplicación, no un servidor aparte). IfcPlusPlus es una librería en C++ similar a IfcOpenShell pero menos activa en los últimos años. Además, Unity Technologies ofrece un producto llamado Unity Reflect que conecta Revit con Unity, pero como hemos visto en la *Tabla 1* es una solución comercial cerrada de pago. Por tanto, xBIM e IfcOpenShell se consideran la combinación óptima para cumplir con los requisitos de interoperabilidad BIM en Unity, minimizando costos.

2.5.1. Elección de librerías para la gestión de datos y geometría IFC

Una vez seleccionadas las tecnologías open source base, se define más concretamente el rol de cada una en el plugin a desarrollar. La librería *xBIM (eXtensible Building Information Modeling)* se empleará principalmente para la **gestión de los datos** del modelo IFC. xBIM Toolkit permite cargar un archivo .ifc y representarlo en memoria como objetos .NET, proporcionando acceso a todas las entidades definidas por el estándar buildingSMART [10]. Ofrece APIs para consultar, crear o modificar elementos del modelo (por ejemplo, obtener todos los IfcWall o cambiar la propiedad Name de un elemento). Esto resulta ideal para implementar en Unity funcionalidades como leer la información alfanumérica de un elemento seleccionado, o aplicar cambios en atributos del modelo BIM desde la interfaz de Unity. Además, xBIM soporta la escritura de nuevos archivos IFC, lo que abre la posibilidad de exportar las modificaciones realizadas de vuelta a un fichero IFC actualizado.

Por otro lado, IfcOpenShell se utilizará para el **procesamiento de la geometría**. IfcOpenShell incluye IfcGeom, un módulo que interpreta las definiciones geométricas paramétricas de IFC (b-reps, CSG, barridos, etc.) y las convierte en geometría explícita (mallas trianguladas) apta para visualizar [11]. Unity requiere meshes trianguladas para renderizar los objetos en escena; IfcOpenShell puede generar dichas meshes directamente a partir del IFC, garantizando una conversión fiel a la intención de diseño original. Esto abarca desde extruir la sección de un muro hasta representar sólidos booleanos complejos. IfcOpenShell **soporta las versiones modernas**

del estándar (IFC2x3, IFC4), por lo que asegura compatibilidad con modelos BIM actuales. En la arquitectura del plugin, xBIM leerá el archivo IFC y proveerá los datos estructurados (árbol de proyecto, propiedades de cada elemento, etc.), e IfcOpenShell se encargará de producir la geometría 3D de cada elemento. De esta manera, se aprovechan las fortalezas de cada librería en su ámbito. En cuanto a la integración técnica, xBIM, al estar en C#, podrá ser utilizada directamente en los scripts de Unity. IfcOpenShell, al ser C++ con API Python, requerirá emplear su ejecutable IfcConvert o integrar un wrapper en C# que invoque sus funciones nativas (se evaluará el método más eficiente).

En términos de rendimiento, interoperabilidad y coste-beneficio, la solución propuesta ofrece **numerosas ventajas**. Al ser open source, no hay costes de licencia y es posible adaptar o depurar el código de las librerías en caso necesario. La interoperabilidad está garantizada al basarse en IFC, pudiendo el plugin cargar modelos exportados de cualquier software BIM. El rendimiento en la carga y visualización dependerá de las optimizaciones que se realicen (por ejemplo, nivel de detalle, manejo de múltiples materiales), pero se parte de librerías diseñadas para manejar modelos complejos de arquitectura en entornos de escritorio. Además, esta arquitectura desacoplada (datos versus geometría) permite escalar a futuros escenarios; por ejemplo, se podría reemplazar IfcOpenShell por otra librería geométrica si fuese necesario, sin afectar al manejo de datos con xBIM. La elección de xBIM e IfcOpenShell proporciona una **base sólida, abierta y comprobada** para el desarrollo del plugin BIM en Unity, alineada con los objetivos de flexibilidad e independencia tecnológica del proyecto.

2.6. Justificación de la elección del motor gráfico Unity 6

Existen diversas plataformas tecnológicas y motores gráficos que podrían emplearse para desarrollar un visor/modificador BIM en tiempo real. Entre las opciones **destacan Unity y Unreal Engine**, los dos motores 3D más populares, además de otros como Godot, Blender Game Engine (en desuso) u opciones especializadas. En este proyecto se ha optado por utilizar Unity 6. A continuación, se expone la justificación técnica de esta elección, comparando Unity con Unreal Engine y valorando cómo satisface los requisitos del entorno BIM propuesto.

2.6.1. Motor 3D vs Software BIM tradicional

Antes de comparar Unity/Unreal, conviene resaltar por qué usar un motor de videojuegos en lugar de simplemente un visor BIM existente. Los motores 3D modernos se han convertido en **herramientas muy valiosas** para la arquitectura y la construcción, más allá de las visualizaciones estáticas. Como señala Morse (2021), "*motores de juego como Unity3D y Unreal se han vuelto herramientas de práctica arquitectónica, no solo para renderizados sino para mostrar datos tridimensionales de edificaciones, interactuar con el mundo físico (escaneos) o usar*

hardware VR". Su capacidad de personalizar la **experiencia de usuario**, conectarse a fuentes de datos externas y aprovechar dispositivos inmersivos es única [14]. Esto significa que con un motor podemos crear aplicaciones a medida: recorridos virtuales inmersivos, simulaciones interactivas (por ejemplo, evacuar un edificio en VR), integraciones con IoT para visualizar estados en tiempo real, etc. Ninguna suite BIM tradicional ofrece tal nivel de **interactividad y programación**. Por tanto, la decisión de usar un motor como plataforma base está motivada por el potencial de innovación y flexibilidad que brinda, esencial para cumplir objetivos como colaboración en VR o Digital Twins.

Ahora bien, *¿por qué Unity y no Unreal Engine?* Ambos motores son técnicamente capaces de visualizar modelos 3D complejos, soportan VR/AR y tienen casos de uso en AEC. La decisión se basó en varios factores:

Unity facilita la **integración de librerías** BIM escritas en .NET/C# (como xBIM) al permitir importar assemblies como plugins y llamar nativamente a sus funciones de lectura IFC; aunque IfcOpenShell es C++, existen wrappers .NET o DLL nativas para Windows. En Unreal, al no ejecutarse sobre .NET, habría que compilar IfcOpenShell para el motor, lo que complica la integración y puede generar incompatibilidades, mientras que en Unity el uso de C# acelera el desarrollo y reduce la curva de aprendizaje y el mantenimiento a largo plazo [14], [15].

En cuanto al rendimiento gráfico, Unreal ofrece **gráficos fotorrealistas** "*out of the box*", pero Unity, configurando su HDRP, alcanza calidades similares en las nuevas versiones (Unity 6) y, al venir más ligero de serie, mantiene altas tasas de FPS con modelos grandes y en XR sin tantos ajustes [16]. El ecosistema de Unity es muy extenso, con multitud de assets, plugins y documentación, y cuenta con soluciones específicas para AEC como Unity Reflect (sincronización de metadatos, streaming de modelos, sesiones multiusuario), lo que demuestra su idoneidad para BIM en tiempo real. Unreal dispone de Datasmith para importación CAD/BIM, pero la comunidad AEC de Unity y proyectos de investigación (por ejemplo, simulaciones de seguridad en obra con un modelo Revit en Unity) son más numerosos [17], [18].

Por último, ambos motores soportan VR y entornos colaborativos, pero Unity lleva ventaja histórica en rapidez de iteración, ofrece integraciones nativas (Oculus, SteamVR) y frameworks (XR Interaction Toolkit) y facilita la red con librerías como Photon o Mirror, permitiendo experiencias multiusuario en VR/AR con Unity Reflect [18].

3. Objetivos

La **falta de interoperabilidad** directa entre modelos BIM y motores gráficos en tiempo real constituye una **barrera tecnológica** relevante para la digitalización avanzada del sector de la construcción. Las soluciones comerciales actuales suelen requerir procesos de exportación intermedios que fragmentan el flujo de trabajo, provocan **pérdida de información y limitan la capacidad de edición colaborativa** en entornos inmersivos. Este proyecto propuesto mejora esta problemática en la carga nativa de modelos BIM modernos, la preservación íntegra de geometría y metadatos, y la modificación dinámica e interactiva en tiempo real.

3.1. Objetivo principal

Ante esta problemática, el presente Trabajo de Fin de Grado tiene como objetivo principal **promover la integración efectiva entre modelos BIM y motores 3D** en tiempo real, mediante el desarrollo de una herramienta propia que elimine estas restricciones. Se persigue demostrar la viabilidad técnica de integrar y manipular información constructiva en un entorno interactivo y multiplataforma, utilizando tecnologías open source que favorezcan la flexibilidad, eficiencia y autonomía en los procesos de visualización y edición de modelos digitales.

3.2. Objetivos específicos

Para alcanzar el objetivo principal, se plantean los siguientes objetivos específicos:

1. **Integración BIM-Unity**: Investigar y lograr la interoperabilidad entre un modelo BIM (en formato IFC) y Unity. Se debe desarrollar un mecanismo que permita a Unity importar el modelo preservando tanto la geometría como la estructura de información (por ejemplo, atributos, relaciones y jerarquía de componentes) de forma íntegra y coherente [1].
2. **Edición de modelos en tiempo real**: Implementar funciones que posibiliten la inspección y edición del modelo BIM dentro de la aplicación Unity en ejecución. Esto incluye la capacidad de seleccionar elementos constructivos y modificar sus atributos (como nombre, material, fase, etc.) o mover y rotar elementos en la escena, con efectos inmediatos en la visualización. El objetivo es alcanzar una experiencia interactiva similar a la que ofrecen los motores de juego modernos [2].
3. **Uso de librerías open source (xBIM/IfcOpenShell):** Seleccionar, aprender a usar e integrar las librerías open source más idóneas para la manipulación de archivos IFC y la generación de geometría en Unity. Se desarrollarán módulos de software que aprovechen xBIM para la gestión de datos y IfcOpenShell para la conversión de

definiciones geométricas en meshes renderizables. Esto permitirá la carga y actualización de modelos BIM manteniendo sincronizados los datos y la visualización [10], [11].

4. **Exportación de cambios de vuelta a IFC**: Implementar, en la medida de lo posible, la capacidad de guardar las modificaciones realizadas en el entorno Unity de nuevo en un archivo IFC actualizado. Este objetivo busca demostrar que los cambios efectuados en la interfaz interactiva pueden reintegrarse al flujo BIM tradicional, permitiendo su reutilización en etapas posteriores del proyecto [10].
5. **Optimización del rendimiento y manejo de la complejidad**: Optimizar la carga y renderización de modelos complejos en Unity. Esto incluye evaluar tiempos de carga, tasa de cuadros por segundo y la eficiencia general de la herramienta. Se explorarán técnicas como la reducción de detalle o la carga diferida de elementos pesados para asegurar una interacción fluida en tiempo real, incluso en modelos de gran escala [2].
6. **Validación mediante caso de estudio**: Probar el plugin desarrollado utilizando uno o varios modelos BIM de ejemplo (preferiblemente proyectos reales o prototipos significativos) para verificar que la solución cumple con las funcionalidades esperadas. La evaluación se basará en la fidelidad de la información cargada, la facilidad de uso de las herramientas de edición en Unity y la compatibilidad de los modelos exportados con los estándares BIM [4].
7. **Análisis de coste-beneficio e impacto en la productividad**: Realizar un estudio comparativo que evalúe el coste de implementación de la solución propuesta frente a las herramientas comerciales existentes. Se analizará el potencial ahorro en costes, la mejora en tiempos de ejecución y el impacto en la productividad de los procesos de diseño y construcción. Este análisis servirá para fundamentar la viabilidad económica y técnica del proyecto [1], [5].

Con estos objetivos, el proyecto no solo busca desarrollar una herramienta funcional y flexible, sino también **generar conocimiento sobre la integración de tecnologías BIM** y motores de juego mediante software open source. La solución propuesta **sentará las bases para futuros desarrollos** en visualización arquitectónica, entornos virtuales de formación y la gestión de obras 4.0, contribuyendo a la innovación en el sector de la construcción ligado a la ingeniería.

4. Análisis del formato IFC

El objetivo de este capítulo es analizar el papel del formato IFC como base estructural para la interoperabilidad en entornos BIM abiertos, así como su relevancia técnica en el desarrollo de soluciones multiplataforma.

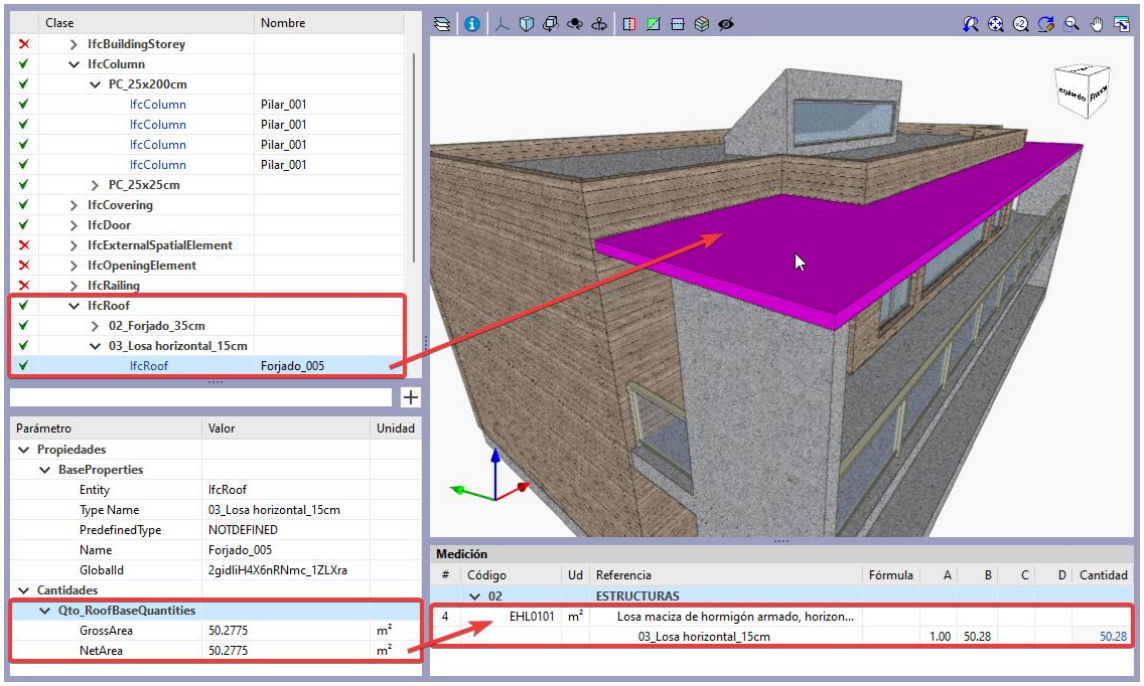

Ilustración 8: "Ejemplo de la clase IfcRoof". Captura de pantalla en Open BIM mostrando un elemento de cubierta (IfcRoof) extraído de un modelo IFC, con sus propiedades de cuantificación (GrossArea y NetArea) y su correspondencia en el listado de mediciones.

El formato IFC (Industry Foundation Classes) es el pilar de los flujos openBIM al ser un estándar abierto, rico y complejo para representar datos de construcción. Técnicamente, **IFC es un esquema de datos** (data schema) neutral y abierto, no controlado por ningún fabricante, cuyo propósito es facilitar la interoperabilidad en la industria de arquitectura, ingeniería y construcción (AEC). Desarrollado y mantenido por buildingSMART International (antes IAI), IFC se publica como estándar internacional ISO (actualmente ISO 16739-1:2024) [19]. En esencia, este estándar **define un modelo de objeto universal** que describe de forma coherente cualquier elemento constructivo y su información asociada en distintas aplicaciones. La *Ilustración 8* muestra, a modo de ejemplo, algunos parámetros predeterminados de un IfcRoof en el mundo BIM.

4.1. Fundamentos matemáticos del IFC

Este apartado tiene como objetivo exponer los **fundamentos matemáticos subyacentes al estándar IFC**, con el fin de comprender su estructura formal, su capacidad de representación geométrica y relacional, y su potencial para el modelado computacional riguroso en entornos BIM. Para ello, se analiza la formalización del modelo mediante teoría de conjuntos, álgebra relacional y estructuras de grafo, así como **las bases geométricas y topológicas** que permiten describir de forma precisa y coherente los elementos constructivos en el espacio tridimensional. Esta visión teórica permite no solo razonar sobre el modelo de datos de forma estructurada, sino también identificar oportunidades para su manipulación algorítmica e integración con motores de renderizado y análisis.

4.1.1. Representación algebraica y teoría de conjuntos

El estándar IFC define un modelo de datos **orientado a objetos**, formalizado mediante el **lenguaje EXPRESS** (ISO 10303-11). En esencia, IFC constituye un modelo de entidad-relación basado en EXPRESS, con cientos de entidades organizadas jerárquicamente en herencia [21]. Matemáticamente, podemos ver un modelo IFC como un conjunto de entidades y relaciones.

Sea $E = e_1, e_2, \dots, e_n$ el conjunto de todas las entidades (elementos) de un modelo. Cada entidad pertenece a una clase del esquema IFC (por ejemplo, IfcWall, IfcColumn) y posee un vector de e_i atributos que describen sus propiedades intrínsecas (dimensiones, nombre, etc.). **IFC asigna a cada entidad un identificador único global** (atributo GlobalId heredado de IfcRoot), lo que garantiza unicidad e integridad referencial en el intercambio de datos. Formalmente, existe una función inyectiva $G: E \rightarrow U$ que mapea cada entidad e_i a un identificador único $G(e_i)$ en el universo U de identificadores, evitando ambigüedades en la referencia de elementos (dos entidades distintas no pueden tener el mismo identificador). Las relaciones en IFC se representan como entidades especiales (subclases de IfcRelationship) que enlazan elementos; por tanto, podemos modelar cada relación como un par ordenado o tupla en términos de teoría de conjuntos.

Por ejemplo, la relación de composición espacial *IfcRelAggregates* puede considerarse un subconjunto $R_{agg} \subseteq E \times E$ tal que $(a, b) \in R_{agg}$ indica que la entidad a agrega o contiene a la entidad b. Muchas de **estas relaciones forman estructuras jerárquicas (árboles), mientras que otras definen grafos más generales**. De este modo, la estructura completa de un modelo IFC puede concebirse como un grafo dirigido donde los nodos son las entidades (con sus atributos) y las aristas son instancias de relaciones que conectan pares de entidades. Esta formalización basada en conjuntos y grafos proporciona una base algebraica para razonar sobre el modelo: por ejemplo, es posible aplicar operaciones de conjunto (uniones,

intersecciones) para filtrar subconjuntos de entidades (como "*todos los muros*" $\subset E$) o componer relaciones transitivamente (obteniendo cadenas de dependencias entre objetos).

4.1.2. Álgebra de conjuntos aplicada

Gracias a la formalidad del lenguaje EXPRESS, **el esquema IFC puede describirse mediante especificaciones algebraicas**. Las entidades (clases) se definen con atributos de ciertos tipos, pudiendo estos a su vez ser otros tipos definidos o selecciones (análogos a un tipo algebraico). El lenguaje EXPRESS permite también restricciones expresadas con lógica (cláusulas WHERE) que actúan como predicados booleanos que deben cumplirse para que una instancia sea válida [24].

En términos de conjuntos, cada clase define un conjunto de instancias posibles y las restricciones especifican condiciones (subconjuntos permitidos). La herencia entre clases implica inclusión de conjuntos: si *IfcWall* hereda de *IfcBuildingElement*, entonces el conjunto de todas las paredes es un subconjunto del conjunto de todos los elementos constructivos. Asimismo, una relación puede verse como una aplicación entre conjuntos; por ejemplo, *IfcRelAssociatesMaterial* relaciona elementos constructivos con materiales, definiendo una función (posiblemente multivaluada) $f:\text{Elementos} \to \text{Materiales}$ en el modelo.

Esta perspectiva de conjuntos permite usar conceptos de álgebra relacional para consultar y manipular la información del modelo IFC. De hecho, **existen enfoques que transforman el modelo IFC a representaciones en grafos** triples (RDF) u ontologías OWL, aprovechando que las entidades IFC y sus relaciones forman un grafo etiquetado, lo que facilita consultas similares a bases de datos graph o relacionales [21].

4.1.3. Modelado geométrico y topológico

IFC soporta una rica descripción geométrica de los elementos, basada en geometría constructiva y teorías topológicas. **La geometría de cada elemento se define mediante una o más representaciones geométricas asociadas** (*IfcShapeRepresentation*), generalmente de tipo tridimensional [19].

Matemáticamente, los objetos geométricos básicos de IFC incluyen puntos, curvas, superficies y sólidos definidos en el espacio euclidiano R^3. Por ejemplo, una entidad *IfcCartesianPoint* representa un punto en coordenadas $(x, y, z) \in R^3$, e *IfcDirection* representa un vector dirección en el espacio. A partir de estos primitivos, **IFC construye geometrías más complejas usando operaciones paramétricas**: una **extrusión** (*IfcExtrudedAreaSolid*) se define por un perfil 2D y un vector de dirección y distancia, generando un sólido barrido linearmente; una **revolución** (*IfcRevolvedAreaSolid*) se define por girar un perfil alrededor de un eje; y las **operaciones booleanas** (unión, intersección, diferencia en *IfcBooleanResult*) permiten combinar sólidos. IFC

también soporta geometría constructiva de sólidos, donde sólidos primitivos (cubos, cilindros, etc.) se combinan mediante operadores booleanos [21].

El modelo geométrico IFC combina **representaciones paramétricas** (barridos, extrusiones), que resultan eficientes y compactas, con representaciones explícitas de B-Rep detallado, más flexibles, aunque complejas, cubriendo desde elementos simples hasta formas orgánicas complejas. Además, gestiona sistemas de coordenadas locales mediante IfcAxis2Placement en 2D y 3D para posicionar objetos de forma relativa, permite anidar estos sistemas dentro de ensamblajes y así asegura la consistencia geométrica en todo el modelo.

4.2. Clases y relaciones en IFC

En este capítulo se examina en profundidad la arquitectura de clases y las relaciones definidas en el esquema IFC, pilar fundamental para la interoperabilidad en entornos BIM. Partiendo de la jerarquía de herencia que arranca en la entidad abstracta *IfcRoot* (proveedora de identificadores globales, historial y metadatos comunes) se describirán las **tres ramas principales del modelo: objetos** (*IfcObjectDefinition*), **relaciones** (*IfcRelationship*) y **definiciones de propiedade**s (*IfcPropertyDefinition*) [23], [24]. A continuación, se detallará cómo cada una de estas categorías articula la representación de elementos constructivos, su vinculación semántica y la extensibilidad del formato mediante conjuntos de propiedades.

4.2.1. Jerarquía de clases

El esquema de IFC organiza sus entidades en una estricta jerarquía de herencia, partiendo de clases abstractas base [23]. La raíz del modelo es *IfcRoot*, clase abstracta de la cual todas las entidades "principales" **derivan directa o indirectamente**. *IfcRoot* proporciona a cada entidad atributos básicos: un *GlobalId* único, información de historial (*OwnerHistory*), nombre y descripción opcionales. A partir de *IfcRoot*, el estándar define tres subclases abstractas fundamentales [24]:

- ***IfcObjectDefinition*** agrupa todas las entidades BIM, físicas o conceptuales, y se divide en *IfcObject* (elementos individuales, por ejemplo, una columna concreta) e *IfcTypeObject* (definiciones de tipo genéricas, por ejemplo, un tipo de columna) [24]. *IfcObject* se especializa en roles semánticos como *IfcActor* para personas u organizaciones, *IfcResource* para materiales, mano de obra y equipos, *IfcProcess* para tareas y cronogramas, IfcControl para costes y requerimientos, IfcGroup para agrupaciones lógicas e *IfcProduct* que engloba los elementos físicos espaciales del edificio, tanto constructivos (*IfcWall*, *IfcBeam*, *IfcDoor*) como territoriales (*IfcSite*,

IfcBuilding, *IfcBuildingStorey*, *IfcSpace*), y todos comparten atributos de colocación espacial (IfcLocalPlacement) y representación geométrica asociada [21].

- ***IfcRelationship**:* Representa las **relaciones objeto-objeto** de forma explícita. IFC sigue un paradigma de relaciones objetivadas, es decir, en lugar de definir una relación como un simple puntero o referencia directa entre dos objetos, se introduce una entidad independiente que encapsula esa relación [27].

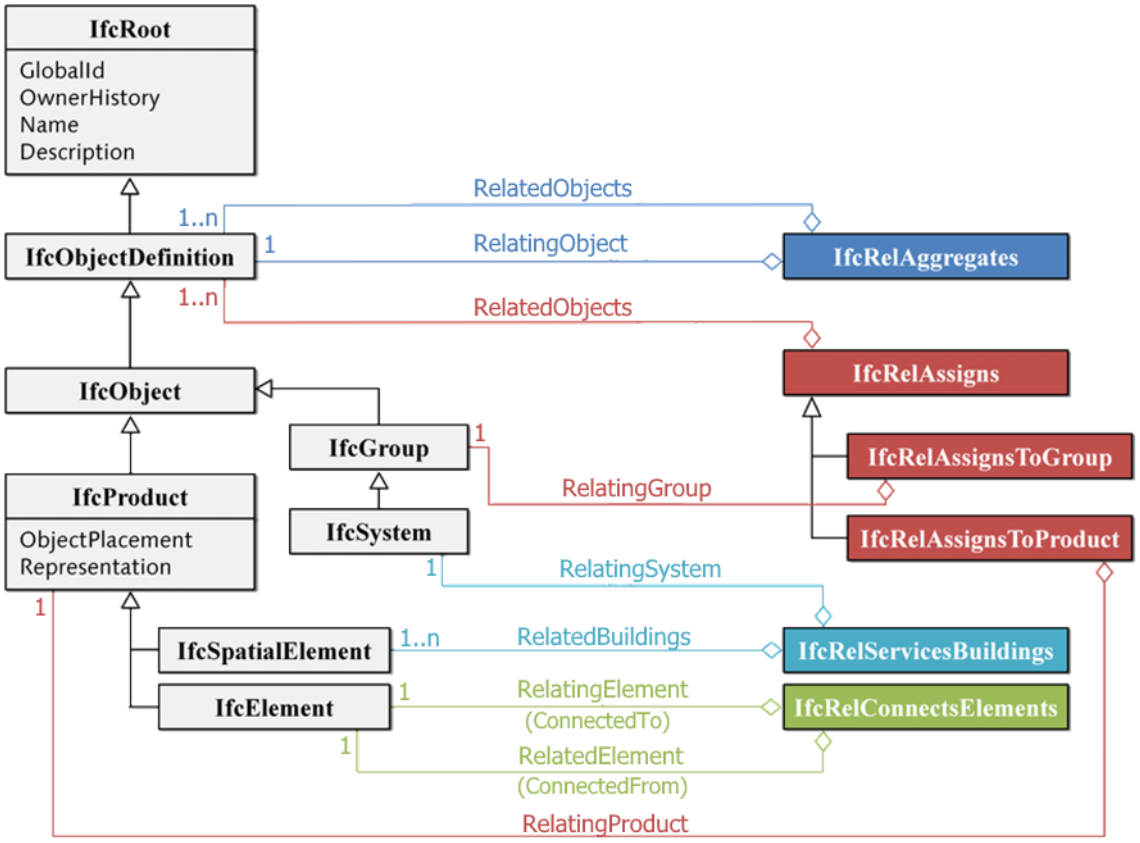

Ilustración 9: "Estructura jerárquica fundamental de IFC". Imagen basada en Kay Smarsly de [27].

Cada instancia de *IfcRelationship* conecta uno o varios objetos (vía atributos como *Relating** y *Related**) y tiene su propia identidad y semántica. Esto permite asignar propiedades a las relaciones y mantener la integridad referencial sin duplicar datos. Las subclases de *IfcRelationship* se organizan en seis tipos fundamentales según la naturaleza de la vinculación:

(1) Composición/Descomposición: *IfcRelAggregates* (y *IfcRelNests*) establecen jerarquías parte-todo, por ejemplo, un edificio agrega varias plantas, o una pared se compone de varios componentes.

(2) Contención espacial: *IfcRelContainedInSpatialStructure*, caso particular de composición espacial, ubicando elementos (muros, columnas) dentro de contenedores espaciales (plantas, edificios).

(3) Asignación: *IfcRelAssigns* asigna recursos o responsabilidades, por ejemplo, asignando una tarea a un actor, o un material a una tarea (aunque para materiales existe una relación dedicada).

(4) Conectividad: *IfcRelConnects* representa conexiones físicas o lógicas, como la conexión entre componentes de instalaciones (una tubería conectada a un equipo) o uniones estructurales (una viga apoyada en una columna). Un ejemplo típico es *IfcRelConnectsPorts* para conectar puertos de instalaciones (red *HVAC*, eléctrica, etc.).

(5) Asociación: *IfcRelAssociates* liga objetos con información externa o adicional, como documentos, clasificaciones o materiales (*IfcRelAssociatesMaterial* vincula un elemento con su definición de material) [12].

(6) Definición: *IfcRelDefines* establece relaciones de definición de propiedades o tipos, por ejemplo, *IfcRelDefinesByProperties* asocia un objeto con un conjunto de propiedades (IfcPropertySet), e IfcRelDefinesByType vincula una ocurrencia (p. ej. una ventana específica) con su tipo genérico (p. ej. tipo de ventana estándar).

Estas relaciones permiten la reutilización de definiciones comunes y la extensión flexible de atributos. Es importante destacar que las relaciones en IFC suelen ser binarias (un objeto relacionado con uno o varios otros), pero algunas admiten conjuntos (por ejemplo, un *IfcRelAggregates* tiene un único objeto principal y una lista de objetos parte). Dado que las relaciones son entidades de primera clase, **IFC define en el esquema atributos INVERSOS en las clases objeto**, que no almacenan nuevos datos, sino que facilitan la navegación inversa del grafo [27].

Por ejemplo, *IfcWall* tiene un atributo inverso *HasOpenings* que lista los *IfcRelVoidsElement* donde ese muro es elemento perforado por alguna abertura; esto permite, a partir de la pared, acceder a sus ventanas/puertas abiertas. Estos atributos inversos se derivan automáticamente de las relaciones existentes, asegurando consistencia y reduciendo redundancia.

- ***IfcPropertyDefinition**:* Se encarga de agrupar todas las entidades relacionadas con la definición y agrupación de propiedades, como *IfcPropertySet*, *IfcPropertySetTemplate* o *IfcComplexProperty*. Esta rama abstracta permite extender la información de los objetos sin alterar directamente su estructura de atributos [24]. Por ejemplo, un muro (*IfcWall*) puede vincularse a un *IfcPropertySet* (entidad contenedora para un conjunto de

propiedades) con propiedades específicas (como su resistencia al fuego o coeficiente de aislamiento) mediante la relación *IfcRelDefinesByProperties*. De este modo, ***IfcPropertyDefinition* actúa como el contenedor conceptual para todas las definiciones de propiedades**, asegurando la extensibilidad y la organización de datos adicionales (p. ej., información de fabricantes, normativas o parámetros de rendimiento).

4.2.2. Estructura de datos y grafos

Internamente, **un modelo IFC puede representarse como un grafo dirigido etiquetado**, donde cada nodo es una instancia de entidad (nodo tipado por su clase IFC) y cada arista es una relación (instancia de *IfcRelationship*) apuntando desde el objeto origen al objeto destino. Esta interpretación gráfica es útil para la implementación en software: muchas librerías BIM manejan el IFC como un conjunto de objetos en memoria con referencias mutuas (punteros), equivalente a un grafo en el que las relaciones actúan como conectores.

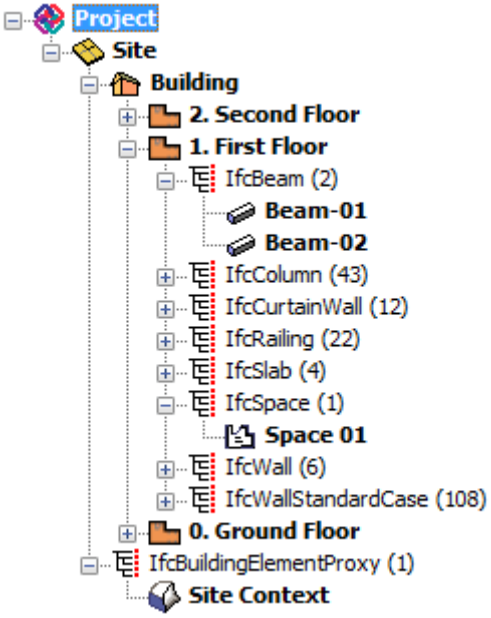

Ilustración 10: "Estructura jerárquica simplificada de un modelo IFC". Capturada del software BIMvision.

Para entenderlo de forma visual se presenta en la *Ilustración 10* se aprecia un IfcProject que contiene un *IfcSite*, éste un *IfcBuilding*, dividido en plantas *IfcBuildingStorey* ("Ground Floor", "First Floor", etc.), las cuales a su vez contienen diversos elementos (muros *IfcWall*, vigas *IfcBeam*, espacios *IfcSpace*, etc.). Esta organización refleja la descomposición espacial del proyecto mediante relaciones de agregación/composición (*IfcRelAggregates*, *IfcRelContainedInSpatialStructure*), formando un árbol jerárquico.

IFC provee un mecanismo extensible para asociar propiedades adicionales a las entidades: los Conjuntos de Propiedades (Property Sets). **Una entidad *IfcPropertySet* actúa como un contenedor nombrado de un conjunto de propiedades individuales (*IfcProperty*)**. Cada *IfcProperty* puede ser de diversos tipos: valor simple (*IfcPropertySingleValue*, con un tipo de dato primitivo como número, texto, booleano, etc.), valor acotado (con mínimo y máximo), lista de valores, tabla de valores (por ejemplo, para curvas de desempeño), enumeración predefinida, o propiedad compuesta (*IfcComplexProperty*).

Esto proporciona un álgebra de propiedades flexible. Por ejemplo, **IFC predefine cientos de PropertySets estándar** (llamados "*Pset_*") para diferentes tipos de objetos; uno común es *Pset_WallCommon* que puede incluir propiedades como *FireRating* (resistencia al fuego), *SoundReduction* (aislamiento acústico), *LoadBearing* (indicador de muro estructural), etc. Un *IfcWall* puede tener adjunto este Pset_*WallCommon*, además de otros.

Esta separación entre la definición de propiedades y las entidades **permite la extensibilidad dinámica del modelo**: nuevos atributos pueden agregarse sin cambiar el esquema base, simplemente definiendo nuevos PropertySets. Por ejemplo, un proveedor podría añadir un *Pset_ManufacturerInfo* a sus componentes con propiedades propietarias (número de serie, garantía, etc.). Gracias a esto, IFC actúa como un meta-modelo capaz de acomodar requerimientos de información adicionales.

4.3. Materiales y atributos en IFC

En IFC, los materiales no son solo textos o propiedades aisladas, sino entidades explícitas que forman parte del modelo semántico. **La clase IfcMaterial representa un material identificado** (por nombre, como "Hormigón Armado C30/37"), al que opcionalmente se le pueden asociar propiedades físicas (densidad, resistencia térmica, etc.) o estilos de representación (color, textura para renderizado). Un elemento constructivo (subclase de *IfcElement*) puede asociarse a uno o más materiales mediante la relación *IfcRelAssociatesMaterial* [13]. Esta relación es un nexo entre la instancia del elemento (p. ej. un *IfcWall* particular) y la definición de material correspondiente, permitiendo una asignación unívoca o compuesta. Matemáticamente, podemos considerar un conjunto M de materiales en el proyecto, donde cada $m \in M$ es una instancia de *IfcMaterial*. La relación de asociación material es entonces una función (o relación $R_{mat} \subset E \times M$) que vincula ciertos elementos $e \in E$ con $m \in M$. IFC soporta varios esquemas para materiales complejos:

- ***Material único**:* El caso sencillo en que un elemento completo usa un solo material (*IfcMaterial*). Por ejemplo, una viga de acero estaría asociada a un *IfcMaterial* "Steel".

- ***Material estratificado (capas)**:* Para elementos compuestos por capas (como un muro multicapa o un techo), se emplea *IfcMaterialLayerSet*. Esta entidad agrega una secuencia ordenada de capas, cada una definida por un *IfcMaterial* y un espesor asignado.
- ***Material por perfil**:* En element os lineales con secciones compuestas (como vigas con núcleo y revestimiento), el empleado en este caso es *IfcMaterialProfileSet*, introducido por IFC4.
- ***Material por componente (constituyentes)**:* Para asociar múltiples materiales a distintas partes de una forma única, está *IfcMaterialConstituentSet*.

Internamente, IFC permite que las definiciones de materiales **sean referenciadas desde varios elementos** (evitando duplicación). Además, es posible anidar materiales (un *IfcMaterial* puede tener como componente otro material, útil en aleaciones o materiales heterogéneos). Todo material puede llevar asociadas propiedades específicas encapsuladas en *IfcMaterialProperties* (por ejemplo, módulo de elasticidad, índice de carbono incorporado, etc.), lo que extiende la caracterización matemática del material más allá de su mera identificación. Es decir, se puede considerar que existe una función de propiedades $P(m)$ para cada material $m \in M$, que devuelve un conjunto de pares atributo-valor (por ejemplo $P(\text{Hormigón}) = \text{densidad} = 2400\ \text{kg/m}^3, \text{f'c} = 30\ \text{MPa}, \ldots$). Esta formalización facilita cálculos automatizados, como estimaciones de volumen de materiales, peso total de acero, etc., extrayendo la sumatoria de atributos de materiales vinculados a los elementos del modelo [13].

4.4. Interoperabilidad y referencia entre datos

En el entorno de la construcción y el modelado de información para la edificación (BIM), **la interoperabilidad y la referencia cruzada** entre datos son pilares esenciales para garantizar la coherencia, trazabilidad y eficiencia a lo largo de todo el ciclo de vida del activo construido. La capacidad de conectar modelos procedentes de distintas disciplinas (arquitectura, ingeniería, instalaciones, sostenibilidad, gestión de obra) y de diferentes herramientas, no solo permite una coordinación más precisa, sino que **habilita una toma de decisiones informada**, colaborativa y basada en datos fiables y actualizados en tiempo real. Sin una interoperabilidad sólida, el flujo de información se fragmenta, se duplican esfuerzos y se multiplican los errores en fases críticas como el diseño, la construcción o la operación. En cambio, cuando los datos están referenciados, normalizados y enlazados adecuadamente, **BIM se convierte en una plataforma de inteligencia constructiva**, donde cada elemento es parte de un ecosistema conectado, rastreable y capaz de evolucionar con el tiempo. Esta visión integrada es la base para avanzar hacia modelos como el gemelo digital o la industrialización de la construcción, donde los datos no solo informan, sino que orquestan **procesos complejos** con precisión y transparencia.

El estándar IFC (ISO 16739-1:2024) [26] utiliza distintos formatos para facilitar el intercambio de información entre aplicaciones. El formato más utilizado es **STEP Physical File (.ifc)**, basado en ISO 10303-21, que, aunque legible y compacto, puede ser lento en modelos grandes. Existen alternativas como IFC XML (.ifcXML), más compatible con herramientas XML, aunque más voluminoso, e IFC comprimido (.ifcZIP), que reduce el tamaño significativamente. Recientemente, se exploran formatos modernos como ifcJSON (para aplicaciones web ligeras) e ifcOWL/RDF (para integración semántica con otras ontologías web). Para más información se recomienda visitar *buildingSmart - Ifc Formats*.

Formato	Sintaxis usada	Uso principal	Ventajas	Desventajas
STEP (.ifc)	ISO 10303-21	Intercambio estándar	Compacto y legible	Lento en grandes modelos
XML (.ifcXML)	XML (ISO 10303-28)	Integración con XML/XPath	Fácil integración genérica	Archivos más grandes
ZIP (.ifcZIP)	ZIP (.ifc comprimido)	Transferencia rápida de datos	Reduce el tamaño en un 60-80%	Requiere descompresión
ifcJSON	JSON	Aplicaciones web ligeras	Simplicidad, interoperabilidad web	Menos extendido
ifcOWL (RDF)	RDF/OWL	Integración semántica	Vinculación semántica robusta	Complejidad técnica

Tabla 2: "Formatos de serialización IFC". Elaboración propia basado en buildingSMART y experiencia propia.

4.4.1. Ejemplo práctico de codificación IFC-STEP

Se presenta un ejemplo de codificación IFC en formato STEP, ya que es el más empleado en la industria, además dado que en este proyecto se trabajará con el archivo "EDIFICIO ONCE NIVELES.ifc" en formato STEP desarrollado por Hiberus Advanced Solutions S.L., es fundamental comprender la **organización de los datos** dentro del archivo IFC y cómo cada entidad se codifica en texto plano.

```
DATA;
#1= IFCORGANIZATION($,'Autodesk Revit 2024 (ESP)',$,$,$);
#5= IFCAPPLICATION(#1,'2020','Autodesk Revit 2024 (ESP)','Revit');
#10 = IFCPROJECT('3y$6AaztT7qfXhqYjEXU28', $, 'EDIFICIO ONCE NIVELES', $, $, $, $, (#20));
#20 = IFCBUILDING('3Qp8fD2oX7$wiO1_T7Bs2A', $, 'Edificio Principal', $, $, $, $, $);
#30 = IFCBUILDINGSTOREY('0S6y$xj9DzAwj8YeKa0LmX', $, 'Planta Baja', $, $, $, $, $);
#40 = IFCWALL('2_f5X5kYv2$fmEjU4HcpjZ', $, 'Muro exterior', $, $, #50, #60, $);
#50 = IFCLOCALPLACEMENT($, #70);
#60 = IFCPRODUCTDEFINITIONSHAPE($, (#80));
#70 = IFCAXIS2PLACEMENT3D(#90, $, $);
#80 = IFCEXTRUDEDAREASOLID(#100, #110, #120, 3.0);
#90 = IFCCARTESIANPOINT((0.0,0.0,0.0));
#100 = IFCRECTANGLEPROFILEDEF(.AREA., $, #130, 0.3, 0.3);
#110 = IFCAXIS2PLACEMENT3D(#90, $, $);
#120 = IFCDIRECTION((0.0,0.0,1.0));
#130 = IFCAXIS2PLACEMENT2D(#140, $);
#140 = IFCCARTESIANPOINT((0.0,0.0));
ENDSEC;
END-ISO-10303-21;
```

Ilustración 11: "Ejemplo de codificación IFC-STEP". Modificación y captura desde Visual Studio Code del archivo 'EDIFICIO ONCE NIVELES.ifc' desarrollado por Hiberus Advanced Solutions S.L.

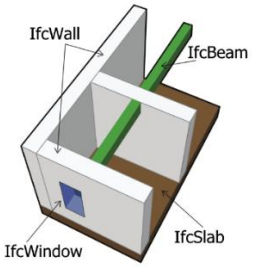

Ilustración 12: "Ejemplo básico de elementos BIM". Muro (IfcWall), viga (IfcBeam), losa (IfcSlab) y ventana (IfcWindow).

Hay una gran cantidad de clases y modelos como se ha visto en la *sección 4.2*, para entender el sistema de referencias utilizado en el formato STEP sobre todo en la geometría se usará como ejemplo un muro IfcWall (#40) indicado en la *Ilustración 11*. Se recomienda su lectura visualizando a la vez la ilustración para su correcta comprensión:

DATA; → Indica el inicio de la sección de datos del modelo IFC.

ENDSEC; → Marca el final de la sección de datos.

END-ISO-10303-21 → Finaliza el archivo STEP.

#1 (IfcOrganization): Define la organización que creó el archivo.

#5 (IfcApplication): Especifica la aplicación usada para generar el modelo.

#10 (IfcProject): Representa el proyecto general.

#20 (IfcBuilding): Define el edificio principal.

#30 (IfcBuildingStorey): Representa una planta del edificio.

#40 → Es un muro (IfcWall) con ubicación espacial (#50) y geometría (#60).

- Ubicación espacial del muro:
  - #50 → IfcLocalPlacement ($, #70);
    - Define la ubicación espacial del muro en el modelo.
    - #70 → IfcAxis2Placement3D (#90, $, $);
      - Define la orientación y posición del muro en 3D.
      - Referencia un punto cartesiano como origen.
      - #90 → IfcCartesianPoint ((0.0, 0.0, 0.0));
        - Indica el punto de origen (0,0,0) donde se coloca el muro en el espacio 3D.
- Geometría del muro:
  - #60 → IfcProductDefinitionShape ($, (#80));
    - Asigna la geometría al muro, referenciando #80, que contiene su representación sólida.
    - #80 → IfcExtrudedAreaSolid (#100, #110, #120, 3.0);
      - Define la geometría del muro como una extrusión.
      - #100 → IfcRectangleProfileDef (.AREA., $, #130, 0.3, 0.3);
        - Define el perfil rectangular de la base del muro.
        - Dimensiones: 0.3 × 0.3 m (ancho y grosor del muro).
        - #130 → IfcAxis2Placement2D (#140, $);
          - Indica la ubicación del perfil 2D antes de extruirse.
          - #140 → IfcCartesianPoint ((0.0, 0.0));
            - Es el punto de origen (0,0) en el plano 2D.
      - #110 → IfcAxis2Placement3D (#90, $, $);
        - Define la ubicación y orientación del perfil en 3D (referencia a #90 → (0.0, 0.0, 0.0)).
      - #120 → IfcDirection ((0.0,0.0,1.0));
        - Indica la dirección de la extrusión (eje Z).
      - 3.0 → Define la altura del muro (extrusión de 3 metros).

5. Metodología

Con el objetivo de abordar de forma estructurada un proceso de desarrollo iterativo, flexible y centrado en la validación continua, **se ha adoptado un enfoque metodológico ágil** basado en la combinación de elementos procedentes de los marcos **Scrum y Kanban**. Esta adaptación ha sido diseñada específicamente para responder a las particularidades del presente proyecto, caracterizado por la **necesidad de incorporar cambios frecuentes, gestionar tareas de alta variabilidad técnica y mantener un control visual riguroso del avance**. La integración de ambos marcos permite equilibrar la planificación incremental propia de Scrum con la fluidez operativa y visualización continua del flujo de trabajo que proporciona Kanban.

Este enfoque se materializa mediante una planificación estructurada en **siete sprints, cada uno con una duración aproximada de 40 horas de trabajo efectivo**, distribuidos en periodos de tres semanas de trabajo por sprint, facilitando un seguimiento claro, organizado y esquemático del avance del proyecto. Además, la metodología ágil utilizada en el equipo de trabajo en la empresa Hiberus ha permitido mantener una comunicación constante y fluida entre los miembros involucrados, así como identificar y resolver oportunamente cualquier posible bloqueo o impedimento, lo que ha resultado en un incremento considerable de la eficiencia y calidad del producto final.

Para gestionar este proceso **se diseñó e implementó un sistema propio basado en un tablero Kanban desarrollado con Node.js y desplegado bajo el dominio *inf.mariogr.com*** (datos de acceso "manuel" – "manuel"). Este tablero Kanban personalizado proporciona una interfaz web accesible mediante credenciales individuales otorgadas a cada uno de los directores del proyecto y al propio alumno, ofreciendo una solución visual intuitiva que facilita el seguimiento y la gestión eficaz de todas las tareas asociadas al proyecto.

Ilustración 13: "Sistema de adición y filtros de tareas". Elaboración propia.

Para añadir una nueva tarea al sistema, es necesario completar un formulario compuesto por varios campos clave. En primer lugar, se debe indicar el título de la tarea, el cual debe ser breve pero lo suficientemente descriptivo como para identificar claramente su contenido. A continuación, se completa el campo de descripción de la tarea, donde se detalla el propósito, alcance y acciones específicas que implica dicha tarea. También se debe especificar el número de horas estimadas que se prevé invertir en su ejecución. Otro campo obligatorio es el del sprint al que se asigna la tarea (por ejemplo, "*Sprint 1*"). La plataforma desarrollada para gestionar las tareas utiliza Node.js y Express, SQLite, Socket.IO y Plesk.

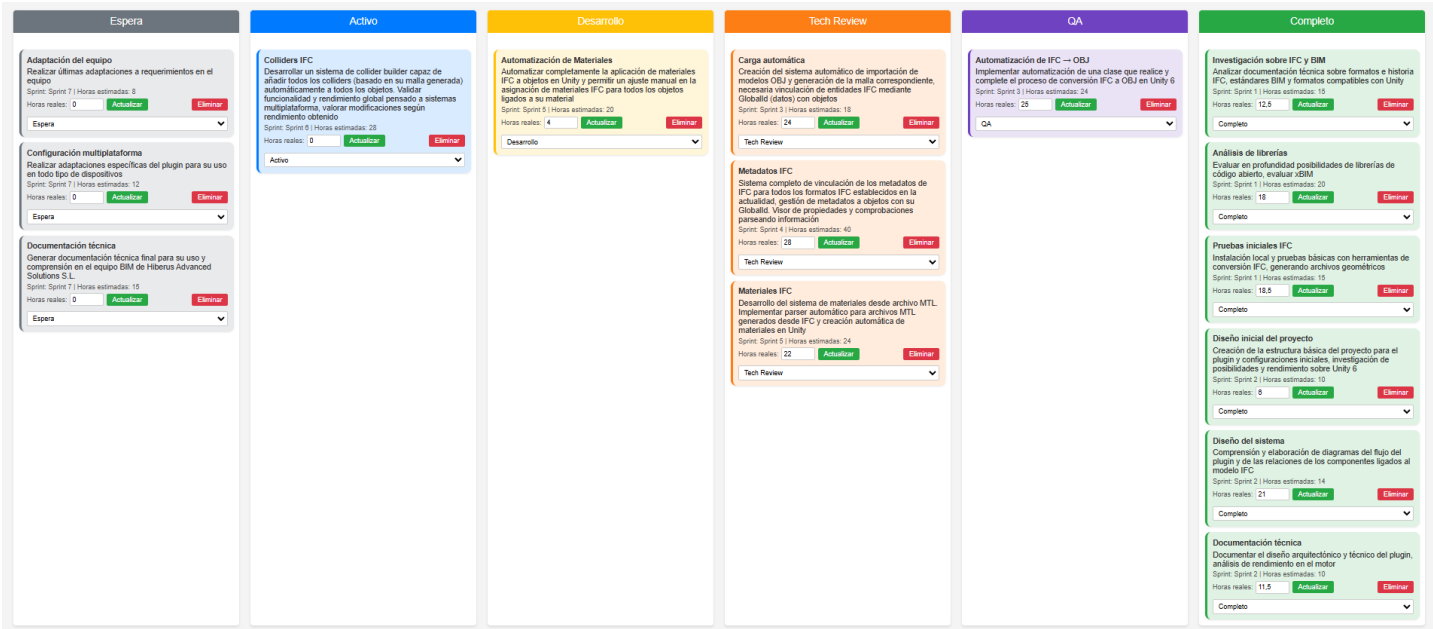

Ilustración 14: "Tablero Kanban basado en Scrum de todas las tareas". Elaboración propia.

El flujo de trabajo utilizado se estructura en columnas claramente definidas que permiten una gestión visual, clara y eficiente del avance del proyecto y el estado actual del desarrollo a simple vista:

- Espera: Las tareas nuevas creadas por parte de los directores se ubican inicialmente aquí, deberán ser aceptadas y aprobadas por el desarrollador y directores.
- Activo: Agrupa las tareas aprobadas y listas para comenzar cuando el desarrollador considere oportuno.
- Desarrollo: Contiene las tareas actualmente en proceso de implementación. Si una tarea no supera las pruebas de calidad (Tech Review o QA), retorna a esta columna para ajustes adicionales basado en el feedback recibido.
- Tech Review: Se realiza una revisión técnica del código y del rendimiento afectado por el desarrollo, validando la adecuación técnica de cada tarea. Expertos en BIM analizarán el comportamiento y rendimiento de la solución.
- QA: Se llevan a cabo pruebas exhaustivas de calidad, corrección de errores y validaciones finales por parte de los directores dirigidas a los requerimientos del propio equipo.
- Completo: Indica que la tarea ha superado satisfactoriamente todas las fases anteriores y se considera totalmente implementada.

Además, cada tarea incorpora además un **sistema de registro de horas reales** que se actualiza constantemente conforme avanza el trabajo. Las horas dedicadas se agregan progresivamente según el desarrollo y estado real de la tarea, permitiendo un control preciso y transparente del tiempo invertido en cada fase del proyecto.

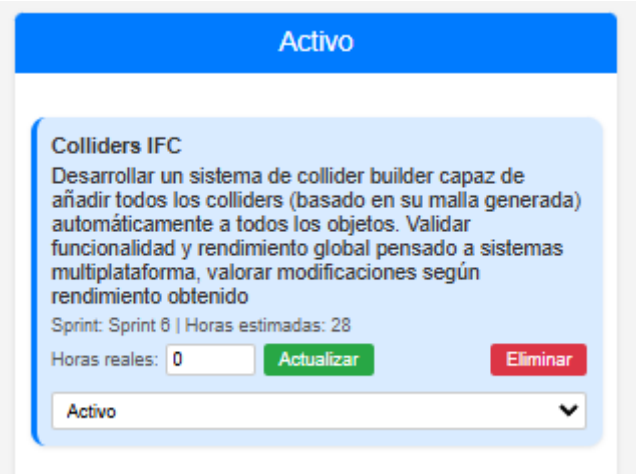

Ilustración 15: "Ejemplo de tarea". La tarea pertenece al Sprint 6, con unas horas estimadas de 28 horas.

Adicionalmente, se realizan **reuniones diarias (dailys)** con una duración aproximada de veinte minutos los días laborables a las 9:00 horas. Estas reuniones permiten la resolución temprana de

bloqueos, facilitan la comunicación directa y efectiva, y favorecen el progreso constante y eficiente del desarrollo.

La Ilustración 14 detalla el desglose de tareas con sus estimaciones y estado (si no visualiza correctamente la información de la ilustración acceda al panel directamente con los *datos de acceso* mencionados), mientras que la *Ilustración 16* ofrece una visión global en formato Gantt para facilitar el seguimiento del progreso y la coordinación del equipo ligada a la carga de trabajo del único desarrollador.

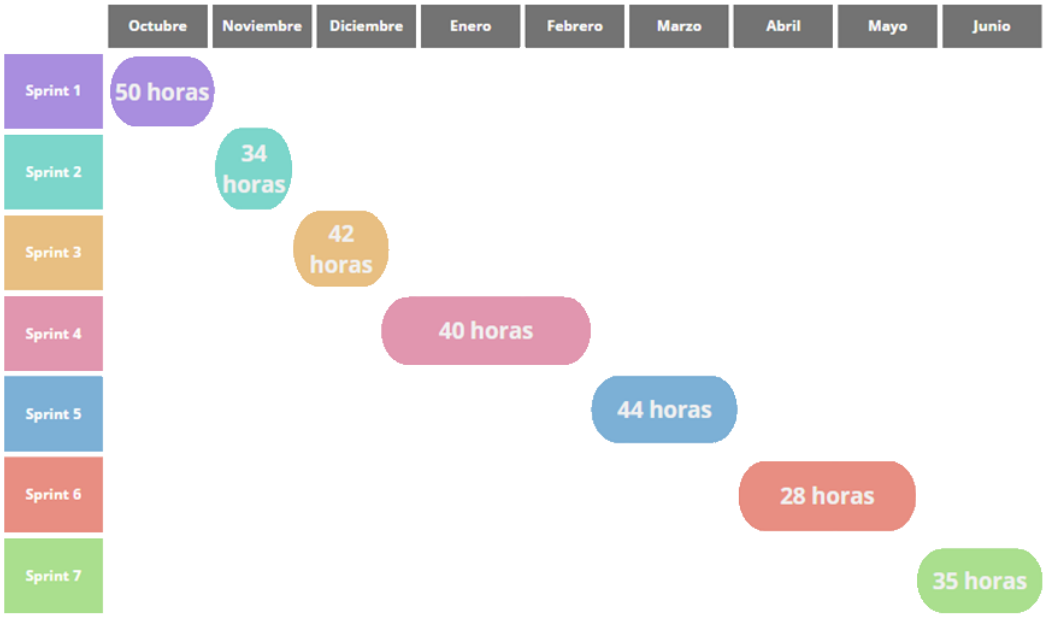

Ilustración 16: "Diagrama de Gantt". Elaboración propia basada en la carga de trabajo de la universidad y festivos locales.

5.1. Herramientas externas utilizadas

Para asegurar un proceso eficiente, ordenado y profesional durante el diseño y desarrollo del proyecto, se han seleccionado cuidadosamente las siguientes herramientas externas, valorando sus beneficios específicos para cada aspecto de este.

Ilustración 17: "Herramientas externas utilizadas". Elaboración propia.

- Unity 6: Elegida como plataforma principal, vea *sección 2.6* para más información
- Librerías IfcOpenShell y xBIM: Elegidas como plataformas de apoyo y guía, vea *sección 2.5* para más información.
- Visual Studio: Utilizado por ser uno de los entornos de desarrollo integrados (IDE) más completos y recomendados para la programación en C#, especialmente por su excelente integración con Unity y sus herramientas avanzadas para depuración y pruebas.
- Git: Seleccionado como sistema de control de versiones debido a su capacidad para mantener un registro exhaustivo del desarrollo, gestionar ramas simultáneas para distintos prototipos o funcionalidades.
- Slack: Implementado como plataforma principal de comunicación instantánea entre los miembros del equipo BIM.
- Microsoft Teams: Utilizado para reuniones formales, presentaciones y seguimientos periódicos del proyecto, aprovechando su capacidad de videoconferencia, gestión de documentos y coordinación eficiente entre todos los involucrados.
- Autodesk Revit: Empleado específicamente para la gestión y visualización avanzada de modelos BIM.

6. Desarrollo del Proyecto

Este capítulo describe de manera sistemática el proceso completo de desarrollo del plugin para Unity, desde la fase inicial de investigación técnica hasta la validación funcional de los resultados obtenidos. Se documenta en detalle **la evolución del** trabajo a lo largo de los diferentes sprints, integrando tanto la fundamentación teórica de las decisiones adoptadas como los aspectos prácticos de implementación y pruebas.

El contenido se organiza en torno a las principales tareas abordadas en cada iteración, incluyendo el estudio del estándar IFC y su aplicabilidad a entornos inmersivos, el análisis de herramientas open source disponibles, la integración con Unity y las estrategias de conversión geométrica. Asimismo, se explicitan las **alternativas tecnológicas** consideradas, los criterios de selección empleados, las dificultades técnicas encontradas y las soluciones propuestas, con el objetivo de justificar cada avance de forma crítica y fundamentada. Finalmente, se incorporan los resultados de las pruebas de rendimiento y estabilidad, así como las lecciones aprendidas, con el fin de orientar la mejora continua de la solución desarrollada y su aplicación a contextos reales.

6.1. Desarrollo del Sprint 1: Investigación y prototipado inicial

El primer sprint del proyecto se planteó como una **fase fundacional**, orientada a establecer los **pilares teóricos y técnicos** que sustentarían el desarrollo posterior del plugin. Su enfoque se centró en explorar en profundidad el estándar IFC, evaluar las principales herramientas open source disponibles para su manipulación y conversión, y llevar a cabo pruebas prácticas preliminares que permitiesen validar la viabilidad técnica del flujo de trabajo propuesto. Esta fase inicial fue especialmente relevante para minimizar los riesgos tecnológicos y garantizar la compatibilidad entre los formatos BIM y el entorno de desarrollo en Unity.

En este apartado se presentan de forma estructurada las tareas realizadas durante el Sprint 1, incluyendo su fundamento teórico, los criterios de evaluación aplicados a las tecnologías consideradas, los resultados obtenidos en las pruebas iniciales y las decisiones adoptadas en función de dichos resultados. Cada tarea se documenta en términos de su planificación temporal, recursos utilizados y aportación al diseño progresivo de la solución. En conjunto, este sprint permitió definir un prototipo técnico funcional que sirvió como punto de partida para los desarrollos de conversión, renderizado y edición que se abordarían en los siguientes ciclos de trabajo. Hay que destacar que previamente se realizó un estudio junto al equipo para valorar qué motor 3D utilizar (vea *sección 2.6*).

El Sprint 1 se concibió como la **fase de cimentación del proyecto**, con tres grandes hitos: la exploración teórica del estándar IFC y su ecosistema BIM, el análisis comparativo de librerías

open source para la manipulación y conversión de modelos IFC, y la validación práctica de la cadena de conversión IFC → OBJ/MTL → Unity.

6.1.1. Tarea 1.1: Investigación sobre IFC y BIM

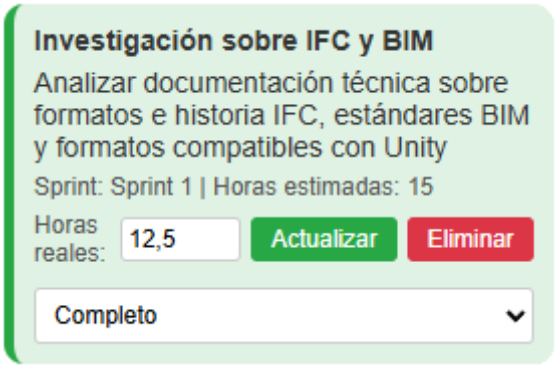

Ilustración 18: Información de la Tarea 1 del Sprint 1 recogida del propio panel de tareas.

Para comprender la compleja estructura de datos del estándar IFC (ISO 16739), se llevó a cabo un **estudio exhaustivo** de la documentación oficial de buildingSMART, manuales académicos, información interna del equipo y ejemplos de modelos sencillos disponibles en el sharepoint del equipo. Este trabajo incluyó un repaso a las entidades geométricas básicas: puntos (*IfcCartesianPoint*), vectores (*IfcDirection*) y perfiles extruidos (*IfcExtrudedAreaSolid*), así como a las representaciones de frontera (*IfcFacetedBrep*, *IfcAdvancedBrep*) y las topologías de *IfcClosedShell* que garantizan la validez de los sólidos. Además, se analizó la **organización jerárquica** de proyectos, edificios y elementos espaciales (*IfcProject*, *IfcBuildingStorey*, IfcLocalPlacement), identificando los nodos de agregación y las relaciones *IfcRelAggregates* y *IfcRelDefinesByProperties* que permiten la reconstrucción del modelo. Gracias a este conocimiento, se definieron las necesidades de parsing y los tipos de datos que deberían exponerse como componentes en el motor de juego, sentando las bases para desarrollar los módulos *IfcEntityLinker*, *IfcProductData* y *IfcPropertySetData*.

Aunque inicialmente se reservaron 15 horas para esta tarea, el análisis concluyó en 12,5 horas. La reducción se atribuye a la claridad de los recursos consultados y al uso de mapas conceptuales para sintetizar rápidamente las secciones más relevantes del estándar incluyendo la documentación aportada en Confluence por el equipo BIM de Hiberus, lo que permitió planificar con mayor precisión los módulos sucesivos.

6.1.2. Tarea 1.2: Evaluación de librerías Open Source

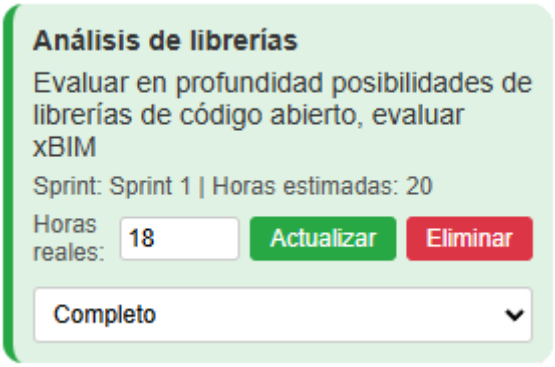

Ilustración 19: Información de la Tarea 2 del Sprint 1 recogida del propio panel de tareas.

La segunda fase del sprint consistió en **comparar las principales librerías** disponibles para trabajar con IFC: xBIM Toolkit, IfcOpenShell, IfcPlusPlus, BIMserver y otras alternativas como IfcPlusPlus e OpenIFCTools. Se definieron tres criterios de selección en orden de importancia:

1. Integración nativa con Unity (C#/.NET).
2. Rendimiento en el procesamiento geométrico (analizado en reuniones explícitas en estimación de rendimientos).
3. Licencia y continuidad de la comunidad y equipo.

Mediante prototipos de carga de modelos y medición de tiempos de parseo, se comprobó que *xBIM* ofrecía una API completa para navegar y modificar la estructura de datos IFC en memoria, mientras que *IfcOpenShell* proporcionaba un motor geométrico robusto (basado en Open Cascade) para generar mallas trianguladas con parámetros ajustables de tessellation y precisión. Las alternativas basadas en Java (*BIMserver*) o con menor actividad de desarrollo (*IfcPlusPlus*) resultaron menos adecuadas para un flujo de trabajo local e integrado directamente en el editor de Unity. Aunque se estimaron 20 horas para esta evaluación, se completaron en 18 horas, dado que el prototipado con IfcConvert ofreció resultados inmediatos y evitó la creación de wrappers adicionales con entornos ya predefinidos en el equipo.

6.1.3. Tarea 1.3: Pruebas iniciales de conversión IFC

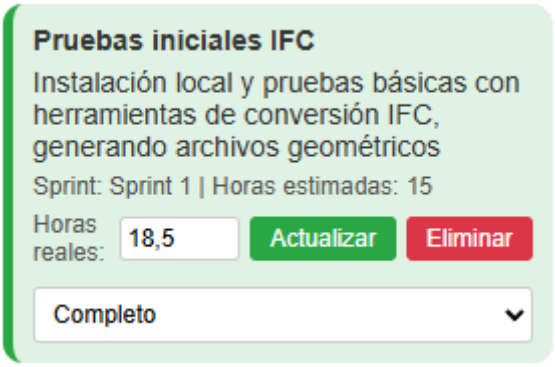

Ilustración 20: Información de la Tarea 3 del Sprint 1 recogida del propio panel de tareas.

Se definió la cadena técnica usando xBIM para metadatos e IfcOpenShell para geometría y materiales y se realizaron las **primeras pruebas** con modelos IFC sencillos. Se evaluaron tres niveles de teselación para garantizar integridad geométrica, compatibilidad de materiales y rendimiento; el nivel medio (modelo de ~5 MB) tardó 3 s en convertirse y mantuvo > 400 fps en URP, mientras que el nivel alto duplicaba el tiempo por solo un 10 % de mejora visual. Al importar faltas de referencias de materiales y elementos lo que requirió 3,5 h extra de depuración y ajustar el *parser* a versiones IFC más modernas, elevando el total a 18,5 h frente a las 15 h previstas.

Con ello, el Sprint 1 sentó una base sólida, definiendo un flujo fiable, minimizando riesgos de incompatibilidad y facilitando la automatización en los sprints siguientes.

6.2. Desarrollo del Sprint 2: Diseño arquitectónico y documentación técnica

Tras la validación técnica inicial realizada durante el primer sprint, el segundo ciclo de trabajo se centró en **definir la arquitectura** interna del plugin y establecer la base documental que guiaría su implementación futura. Esta fase tuvo un papel estratégico dentro del desarrollo, ya que su correcta ejecución permitiría consolidar una estructura de componentes sólida, escalable y mantenible, evitando improvisaciones técnicas durante las siguientes etapas del proyecto.

El trabajo se organizó en tres bloques fundamentales: la creación de la estructura base del proyecto en Unity, el diseño detallado del sistema y sus principales componentes mediante diagramas y especificaciones funcionales, y finalmente, la documentación técnica exhaustiva destinada a facilitar la colaboración y futuras extensiones por parte del equipo BIM de Hiberus. Cada una de estas tareas fue planificada cuidadosamente, evaluando el uso de patrones de diseño, el rendimiento del sistema y la claridad del flujo de ejecución dentro del motor. A continuación, se presentan los resultados obtenidos en cada una de las fases, incluyendo las decisiones técnicas adoptadas y los aprendizajes derivados del proceso.

6.2.1. Tarea 2.1: Diseño inicial del proyecto

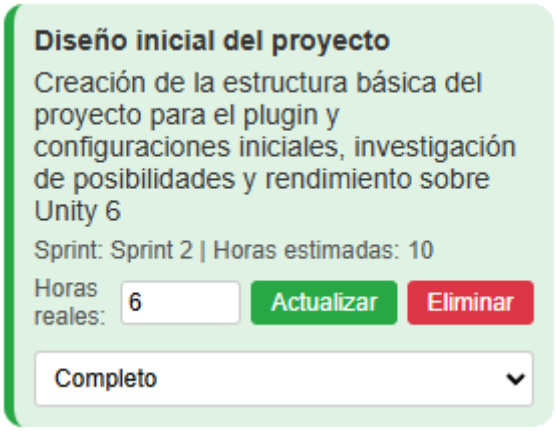

Ilustración 21: Información de la Tarea 1 del Sprint 2 recogida del propio panel de tareas.

Durante las seis primeras horas se estableció la estructura de carpetas y los módulos básicos de Unity, incluyendo:

Un Assembly Definition (*IfcUnityEditorPlugin.asmdef*) para separar el código de Editor del Runtime, mejorando los tiempos de compilación y evitando referencias circulares.

La carpeta Scripts/Core, con *Config.cs* para parametrizar rutas (*IfcConvert*, carpetas de importación) y *HelperFunctions.cs* para utilidades comunes (gestión de archivos en *StreamingAssets*, refresco de *AssetDatabase*).

En Scripts/Initialization, la clase *ModelInitializer.cs* que orquesta la carga automática de un IFC al abrir la escena, creando el GameObject raíz y asignando el componente *IfcFileAssociation*.

En Editor/MenuEntries.cs, los menús "IFC → Cargar archivo IFC" y "IFC → Cargar desde URL", delegando en IfcFileLoader el lanzamiento del ejecutable de conversión y la posterior importación de malla y metadatos.

Este paso fue más ágil de lo previsto gracias a la investigación previa y diseño conjunto con el equipo.

6.2.2. Tarea 2.2: Diseño del sistema

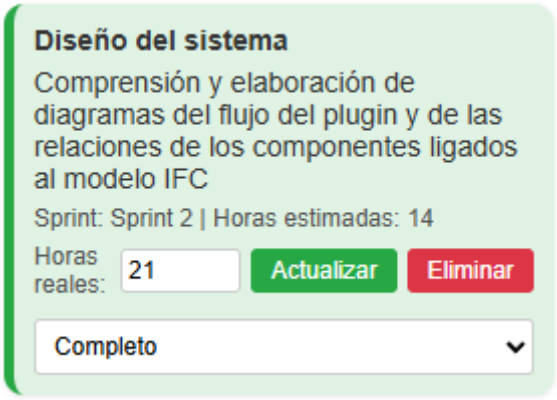

Ilustración 22: Información de la Tarea 2 del Sprint 2 recogida del propio panel de tareas.

La fase de diagramado y modelado de componentes se alargó a 21 h debido al nivel de detalle exigido:

Se emplearon herramientas como draw.io para elaborar borradores de diagramas de flujo que describen, paso a paso, el recorrido desde la invocación del menú hasta la aplicación final de creación de colliders y materiales en cada elemento de la entidad IFC.

Un Diagrama de Componentes muestra la interacción entre *IfcFileLoader*, *IfcEntityLinker*, *MaterialParser*, *ColliderBuilder* y *UnityModelChecker*. En él se definieron claramente las responsabilidades de cada clase:

- IfcFileLoader prepara los OBJ/MTL y crea el prefab base (con IfcFileAssociation).
- IfcEntityLinker reparte la información de xBIM a IfcProductData, IfcTypeData y reconstruye la jerarquía espacial.
- MaterialParser y MaterialMap traducen las definiciones de .mtl a Material de Unity;
- ColliderBuilder añade los colliders óptimos según la clase IFC;
- UnityModelChecker valida en tiempo de edición la integridad del modelo frente a posibles cambios.

Se documentó en detalle para el equipo el ciclo de vida y funcionamiento para posibles modificaciones a futuro. La complejidad de hecho quedó en el punto de documentación formal para el equipo detallando el flujo que debe tener.

6.2.3. Tarea 2.3: Documentación técnica

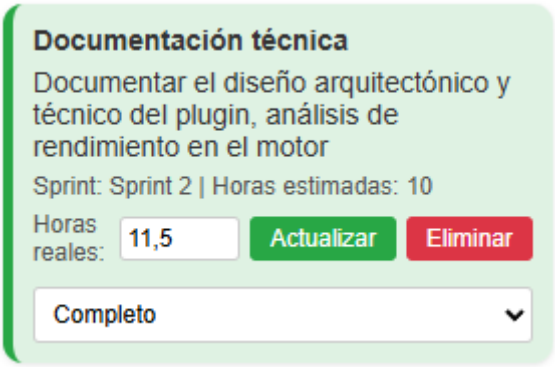

Ilustración 23: Información de la Tarea 3 del Sprint 2 recogida del propio panel de tareas.

Se inició en Confluence una Guía de Arquitectura destinada a facilitar la **evolución y el mantenimiento** del plugin por parte del equipo BIM. En ella se describen los patrones de diseño aplicados: *Singleton* en la clase *Config* para garantizar un acceso único a los parámetros globales; *Factory* en *MaterialParser* para abstraer la creación y configuración de materiales en Unity a partir de archivos .mtl; y *cadena de responsabilidad* en *UnityModelChecker* para encadenar validadores de integridad IFC, aplicación de materiales y consistencia de la jerarquía espacial.

Además, se llevó a cabo un análisis de rendimiento a nivel de módulos unitarios. Se compararon las tasas de fotogramas al importar un modelo IFC de referencia (aprox. 5 MB) en URP sin colliders dinámicos y se midió la memoria consumida por las mallas antes y después de aplicar optimizaciones. Este estudio reveló una sobrecarga de memoria por acumulación de buffers, y el tiempo extra se dedicó a investigar su causa.

6.3. Desarrollo del Sprint 3: Automatización de la conversión y carga

Tras definir la arquitectura del sistema en el sprint anterior, el tercer ciclo de trabajo se centró en **automatizar el proceso de conversión y carga** de modelos IFC en Unity, con el objetivo de eliminar por completo la intervención manual del usuario y permitir un flujo continuo y autónomo dentro del entorno de desarrollo. Esta fase supuso un avance fundamental hacia una solución completamente integrada, en la que tanto la generación de geometría como la asociación semántica con los metadatos BIM se realizasen de forma transparente para el usuario final.

El trabajo se estructuró en torno a dos tareas principales: la automatización de la conversión de modelos IFC a formato OBJ/MTL mediante la herramienta *IfcConvert*, y la carga automática de dichos archivos en Unity, vinculándolos de forma precisa con su información de origen a través del identificador único *GlobalId*. Aunque el planteamiento inicial era relativamente simple desde el punto de vista funcional, la implementación práctica implicó resolver múltiples desafíos

técnicos, especialmente en lo relativo a la asincronía del flujo de eventos en el editor de Unity, la robustez del tratamiento de errores y la coherencia de la jerarquía resultante. A continuación, se detallan ambas tareas, incluyendo su planificación, complejidad, resultados obtenidos y lecciones aprendidas.

6.3.1. Tarea 3.1: Automatización de IFC → OBJ

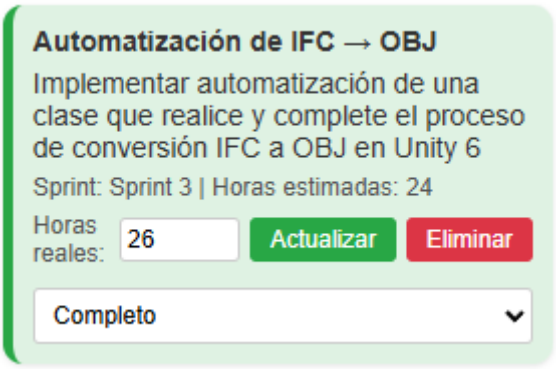

Ilustración 24: Información de la Tarea 1 del Sprint 3 recogida del propio panel de tareas.

Se partió de la clase *IfcFileLoader*, que hasta ese momento invocaba IfcConvert.exe de manera estática, y se la rediseñó para gestionar todo el ciclo de conversión de principio a fin de forma asíncrona y controlada desde una ventana personalizada del Editor de Unity. El objetivo fue ofrecer al usuario una interacción fluida, donde pudiera lanzar la conversión y ver en tiempo real el **progreso** sin bloquear el hilo principal de la interfaz, esto ayudó en gran medida en debuggear aspectos y cargas de la clase.

Para ello, se implementó una barra de progreso de tipo ASCII para evaluar el porcentaje de carga del modelo. Conscientes de que pueden producirse errores de E/S (por ejemplo, si el fichero IFC está bloqueado) o geometrías inválidas en la salida, se añadió un mecanismo de reintentos y eliminación de información adherida al objeto en cuestión con el fin de continuar el proceso con el menor de los problemas posible.

Una vez completada la conversión, es necesario sincronizar los nuevos archivos OBJ/MTL con el proyecto de Unity. Para ello se llama a *AssetDatabase.Refresh()*, y a continuación se espera a que los ficheros estén realmente disponibles.

Aunque en un principio se asignaron 24 horas a esta tarea, el esfuerzo finalmente ascendió a 26 horas. Aproximadamente 4 horas se dedicaron a depurar los interbloqueos entre el hilo de la ventana de progreso y los *callbacks* del *AssetDatabase.Refresh*, y 2 horas adicionales se emplearon en robustecer el tratamiento de errores de *IfcConvert* cuando la malla generada contenía geometrías corruptas (polígonos degenerados, normales faltantes, etc.). Este tiempo

extra ha sido crucial ya que se ha conseguido minimizar al máximo estas geometrías corruptas que provocan las propias aplicaciones al exportar en IFC.

6.3.2. Tarea 3.2: Carga automática

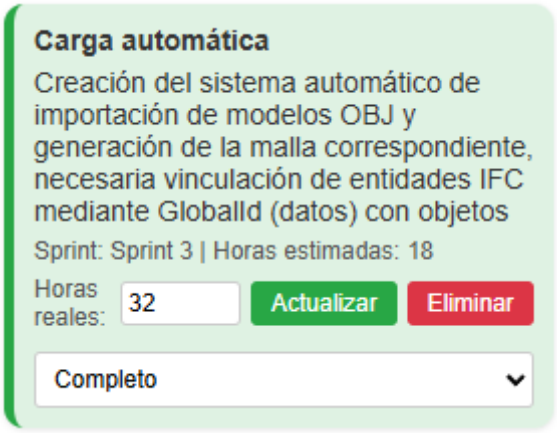

Ilustración 25: Información de la Tarea 2 del Sprint 3 recogida del propio panel de tareas.

Una vez que la conversión a OBJ se realiza sin intervención, el siguiente reto consistió en que Unity importe estos archivos de **forma inmediata y los vincule** con sus datos IFC usando el *GlobalId* propio y único de cada uno de los elementos que conforman el modelo. Para ello se diseñó la clase *ModelInitializer*, la cual se suscribe al evento *EditorApplication.update* y, en cada tick, monitoriza la carpeta *StreamingAssets* detectando nuevos ficheros .obj. Paralelamente, se implementó un *AssetPostprocessor* personalizado mediante *OnPostprocessAllAssets*, encargado de crear automáticamente un GameObject provisional por cada malla importada y de añadirle un componente *ObjMetaData* con la ruta del *asset* y el *GlobalId* extraído del nombre del fichero.

Acto seguido, se invoca *IfcEntityLinker*, que recorre el *IfcStore* buscando coincidencias de *GlobalId* para asociar cada *GameObject* a su *IfcProductData* e *IfcPropertySetData*, **reconstruir la jerarquía espacial** (Proyecto → Sitio → Edificio → Planta) para poder aplicar los colliders y materiales que se gestionarán en el futuro de la forma más rápida y sencilla posible. Para garantizar la robustez del proceso, se añadieron pruebas unitarias que simulan la importación de un modelo sencillo (*IfcWall*) de ejemplo para una carga sencilla y un debug instantáneo, evitando el retraso del modelo anterior (EDIFICIO DE ONCE NIVELES.ifc).

Aunque inicialmente se asignaron 18 horas para completar esta tarea, finalmente se emplearon 32 horas debido a varios retos asociados al. En primer lugar, la sincronización asíncrona entre la detección de nuevos .obj en *ModelInitializer.Update()* y la ejecución de *OnPostprocessAllAssets()* requirió afinar repetidamente el uso de *EditorApplication.delayCall* y ciclos de comprobación tras *AssetDatabase.Refresh()*, para asegurarnos de que Unity había importado por completo cada

malla antes de proceder al enlace. Además, se intentó mejorar el problema de memoria que se tenía desde el sprint anterior minimizando los daños.

Finalmente, para evitar "*entidades huérfanas*" diseñamos en *UnityModelChecker* un validador que compara el número total de instancias en el *IfcStore* (vía IfcStore.Instances.Count) con la cantidad de componentes *IfcProductData* presentes en la jerarquía de la escena. Cuando detecta discrepancias, lanza un *Debug.LogError* en la consola del Editor, indicando qué *GlobalId* faltan o están duplicados.

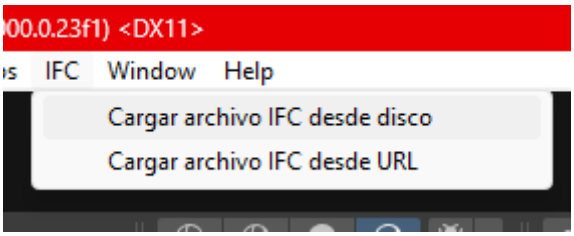

Ilustración 26: "Guía de carga de modelo IFC". Captura tomada de Unity 6.

Gracias a estos ajustes el usuario puede pulsar en el menú "IFC → Cargar archivo IFC desde disco o desde URL" y, en pocos minutos acorde al nivel de geometría y metadatos, obtener en la escena un modelo completamente importado, vinculado y jerarquizado sin intervenciones manuales.

6.4. Desarrollo del Sprint 4: Sistema de Metadatos

Tras haber automatizado satisfactoriamente la conversión geométrica y la carga jerarquizada de los modelos IFC en Unity, el cuarto sprint se centró en profundizar en la **integración de los metadatos** alfanuméricos asociados a los elementos constructivos, un componente esencial para la explotación semántica de los modelos BIM. Este tipo de información (que incluye propiedades físicas, funcionales, normativas y de mantenimiento) resulta clave para dotar de valor operativo a la representación geométrica y facilitar su uso por parte de técnicos, gestores y herramientas externas.

El objetivo principal de este sprint fue construir un **sistema completamente automatizado** de extracción, enlace y visualización de los conjuntos de propiedades (*Property Sets*) definidos en los modelos IFC, con independencia de la versión del estándar utilizada (Ifc2x3, Ifc4 o extensiones personalizadas). Para ello, se reforzaron los mecanismos de enlace entre entidades, se desarrollaron inspectores personalizados que permiten explorar los metadatos desde el propio Editor de Unity y se integraron validaciones de integridad para garantizar la coherencia del modelo. Esta fase marca un hito importante, ya que convierte a Unity no solo en un visualizador

de modelos geométricos, sino también en un entorno plenamente consciente de la información estructural del modelo BIM.

6.4.1. Tarea 4.1: Metadatos IFC

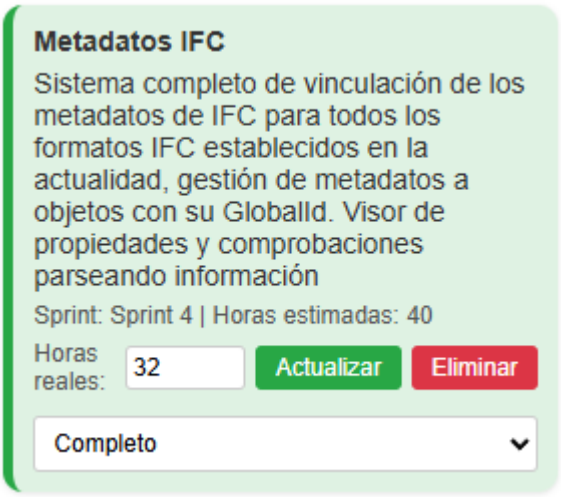

Ilustración 27: Información de la Tarea 1 del Sprint 4 recogida del propio panel de tareas.

Se amplió *IfcEntityLinker* para soportar cualquier estándar IFC (Ifc4.x, Ifc5 y sus extensiones): tras reconstruir la geometría y la jerarquía espacial, el método *AddPropertiesOfProduct* recorre todas las relaciones *IfcRelDefinesByProperties* de cada *IfcProduct* y crea dinámicamente componentes *IfcPropertySetData* que agrupan objetos *IfcPropertyData*. Cada *IfcPropertyData* extrae nombre, valor nominal, unidad y tipo de dato (*IfcLabel*, *IfcReal*, *IfcLogical*…) mediante utilidades de *parseo*, garantizando compatibilidad entre variantes IFC.

Para facilitar su inspección y edición en el Editor de Unity se desarrollaron inspectores personalizados en *IfcPropertySetDataDrawer.cs* e *IfcPropertyDataDrawer.cs*, junto a *ReadOnlyDrawer.cs*, que muestran listas plegables de propiedades, permiten filtrar por nombre de PSet o palabra clave y exportar todo en CSV. Además, se integró un validador en UnityModelChecker que compara el número de *IfcPropertyData* instanciados con el conteo de propiedades en el *IfcStore* de xBIM y registra en consola errores con *GlobalId* y nombre del *PSet* afectado.

Con este desarrollo ya existe un flujo completamente automático que importa **geometría y metadatos IFC**, los **presenta en el Editor** de Unity y asegura su **consistencia**. El siguiente gran paso será implementar la gestión de materiales y el parseo MTL para adaptarlos al motor.

6.5. Desarrollo del Sprint 5: Materiales

Una vez incorporados con éxito tanto la geometría como los metadatos IFC al entorno de Unity, el quinto sprint se centró en la **gestión avanzada de los materiales**. En el modelado BIM, los materiales no solo cumplen una función estética, sino que también **aportan información visual** clave sobre el comportamiento constructivo, **propiedades físicas** y acabados. Por ello, la correcta interpretación y aplicación de los materiales definidos en los archivos MTL (generados durante la conversión IFC → OBJ) constituye una parte esencial para lograr una representación fidedigna y operativa del modelo.

El objetivo principal de este sprint fue implementar un subsistema completo de lectura, creación, trazabilidad y asignación automatizada de materiales, que ofreciera tanto robustez en su funcionamiento como flexibilidad para adaptaciones manuales por parte del usuario. Este trabajo se organizó en dos grandes tareas: el desarrollo de un parser especializado para archivos MTL, capaz de generar materiales URP/Lit en Unity a partir de parámetros físicos reales, y el diseño de un sistema automático de asignación y mapeo, basado en etiquetas IFC, que aplicara dichos materiales de forma coherente sobre la geometría importada. A continuación, se detallan ambas fases, incluyendo los criterios técnicos seguidos, los retos encontrados y los resultados alcanzados.

6.5.1. Tarea 5.1: Materiales IFC

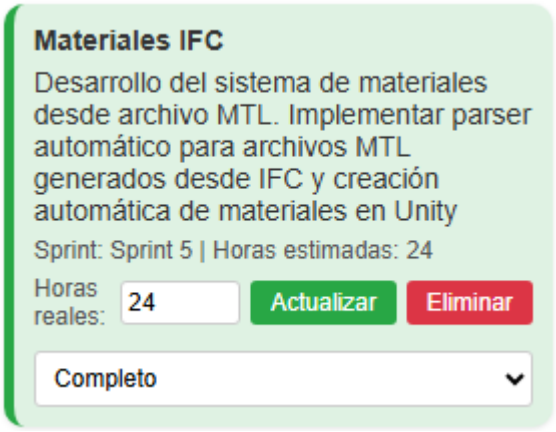

Ilustración 28: Información de la Tarea 1 del Sprint 5 recogida del propio panel de tareas.

Para soportar cualquier archivo .mtl producido por *IfcOpenShell* previamente mediante la cadena IFC → OBJ/MTL, se implementó la clase *MaterialParser* encargada de la gestión de captura y creación de materiales.

```
2   newmtl surface-style-870-tierra
3   Kd 0.380392156862745 0.294117647058824 0.243137254901961
4   Ks 0.5 0.5 0.5
5   Ns 128
6   newmtl surface-style-42774-monocapa-crema-claro
7   Kd 0.709803921568627 0.682352941176471 0.607843137254902
8   Ks 0.5 0.5 0.5
9   Ns 64
10  newmtl surface-style-47322-ladrillo-cer--mico-macizo
11  Kd 0.537254901960784 0.470588235294118 0.4
12  Ks 0.5 0.5 0.5
13  Ns 64
14  newmtl surface-style-45653-pintura-blanca
15  Kd 1 1 1
16  Ks 0.5 0.5 0.5
17  Ns 64
```

Ilustración 29: "Información de archivo MTL". Captura tomada del propio archivo EDIFICIO DE ONCE NIVELES.mtl con Visual Studio Code.

El *parser* recorre el archivo MTL línea a línea, omitiendo comentarios y espacios, y extrae los parámetros *Kd* (color difuso), *Ks* (reflectividad) y *Ns* (rugosidad) de cada bloque *newmtl*. Con esos valores crea instancias de Material empleando el shader URP/Lit, asignando *albedoColor*, *metallic* y *smoothness* según los datos extraídos. Al importar el MTL del edificio de once niveles, con sus 120 materiales, se generaron automáticamente en Assets/IfcMaterials/{nombreModelo} 120 materiales nombrados según su etiqueta de superficie, en un proceso prácticamente instantáneo. Además, el parser detecta y reporta errores de sintaxis que provocan errores en la vinculación para tratar de minimizar los problemas.

Para mantener la trazabilidad con el modelo IFC, cada material se asocia a un componente *IfcUnityMaterialLink* que almacena el *GlobalId* original, lo que permite tanto su aplicación automática como la edición manual en el Inspector.

6.5.2. Tarea 5.2: Automatización de Materiales

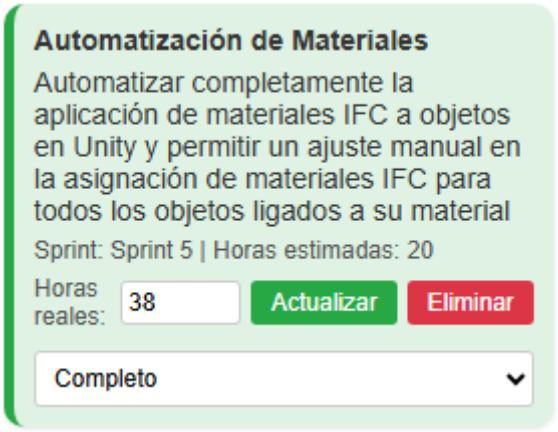

Ilustración 30: Información de la Tarea 2 del Sprint 5 recogida del propio panel de tareas.

En esta fase se completó el sistema que, a partir de los materiales generados por el *MaterialParser*, asigna automáticamente los *Material* de Unity a cada *MeshRenderer* vinculado a un *IfcProductData*, manteniendo al mismo tiempo una interfaz de usuario para ajustes manuales por requerimiento del equipo.

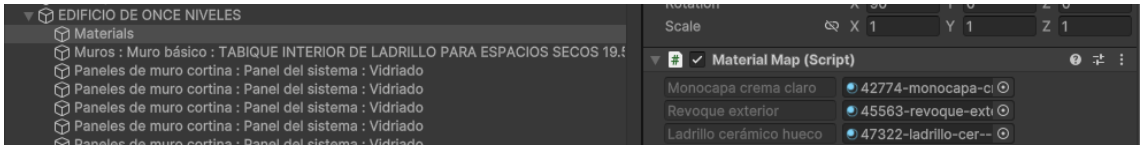

Ilustración 31: "Ejemplificación de Materiales en Unity". Captura tomada de Unity 6.

En el GameObject raíz (EDIFICIO DE ONCE NIVELES) se añade el componente *MaterialMap* que contiene una lista de *MaterialMapKeyValue*. Cada entrada asocia una etiqueta IFC (*SurfaceStyleLabel*) con un asset Material de Unity y, al inicializarse, construye internamente un diccionario: *Dictionary<string, Material>* que usa para iterar por todos los *MeshRenderer* hijos y, si el IfcProductData.*SurfaceStyleLabel* coincide con una clave, aplicar el apropiado *renderer.sharedMaterial*.

El inspector muestra una sección personalizada "Material Map" (vea la *Ilustración 32*) con cada par (etiqueta IFC ↔ material Unity). El **usuario puede modificar el material** de un tipo aplicando texturas personalizadas o shading de alto rendimiento para renderers con el objetivo de comercialización del producto.

Este diseño como diccionario editable permite cubrir casos en que varios elementos (por ejemplo, muros y paneles de vidrio) comparten un mismo material conjunto, así como adaptarse a proyectos con caracteres especiales o nombres en castellano mediante normalizaciones y además personalizar el modelo a según el caso necesario: desde optimización en VR o simplemente como visor de datos a rendering profesional y comercial.

6.6. Desarrollo del Sprint 6: Sistema de Colliders

Una vez completados los subsistemas de geometría, metadatos y materiales, el sexto sprint se centró en un componente clave para habilitar la interacción física en entornos 3D: la **generación automatizada de colliders**. En el contexto de aplicaciones BIM inmersivas y simulaciones interactivas, los colliders permiten que los elementos del modelo reaccionen correctamente ante eventos físicos, como colisiones, navegación o validaciones espaciales. Su incorporación es fundamental para garantizar que los modelos no solo sean visualmente correctos, sino también funcionales en tiempo real.

El objetivo de este sprint fue diseñar un sistema capaz de identificar, clasificar y asignar *colliders* de forma automática a los objetos IFC importados, optimizando el rendimiento del entorno de ejecución y permitiendo un alto grado de personalización. Para ello, se desarrolló un componente especializado (*ColliderBuilder*) que analiza la jerarquía del modelo, reconoce las clases IFC relevantes y genera colliders ajustados a la geometría de cada elemento. A continuación, se detallan las estrategias implementadas, las optimizaciones introducidas y los mecanismos de validación aplicados para asegurar la escalabilidad y el buen comportamiento del sistema en diferentes plataformas y contextos de uso.

6.6.1. Tarea 6.1: Colliders IFC

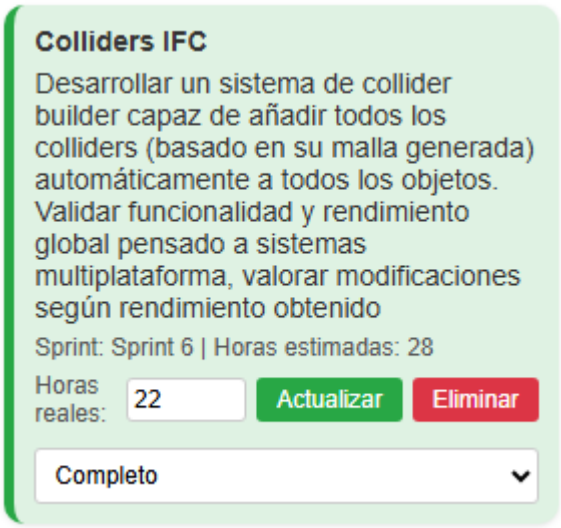

Ilustración 32: Información de la Tarea 1 del Sprint 6 recogida del propio panel de tareas.

La clase *ColliderBuilder* automatiza la creación de colliders en un modelo BIM importado desde IFC recibiendo el *GameObject* raíz y ejecutando dos pasos clave: primero, asegura que existan en el proyecto tres capas unificadas (Walls (29), Grounds (30) y Doors (31)); luego llama a *BuildCollidersRecursive(building)*, que recorre toda la jerarquía y, en cada nodo sin hijos con componente *IfcProductData*, usa *HasCollider(ifcClassName)* para comparar su clase IFC con las listas predefinidas de muros, suelos y puertas, asignándole la capa correspondiente o descartándolo si no encaja. Cuando se requiere *collider*, añade un *MeshCollider* al *GameObject* a partir de su propia malla (identificada por su *GlobalId*), simplificando la física del modelo y permitiendo filtrar colisiones por capa (por ejemplo, excluir puertas o suelos según necesidades).

6.7. Desarrollo del Sprint 7: Adaptaciones, compatibilidad y documentación

El séptimo y último sprint se centró en consolidar y preparar el plugin para su **despliegue en entornos** productivos: adaptar la herramienta a los requisitos funcionales y de idioma del equipo

BIM de Hiberus Advanced Solutions S.L., garantizar su compatibilidad multiplataforma (escritorio y móvil) y completar la documentación técnica definitiva en Confluence para usuarios finales y desarrolladores futuros. Con los módulos ya validados, esta fase cerró el ciclo iterativo de desarrollo, reforzando la robustez, portabilidad y sostenibilidad del software según las necesidades operativas de Hiberus y las mejores prácticas BIM.

6.7.1. Tarea 7.1: Adaptación del equipo

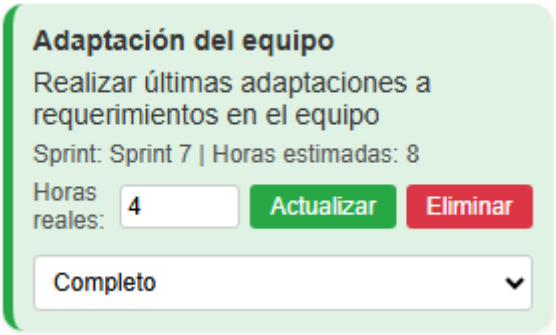

Ilustración 33: Información de la Tarea 1 del Sprint 7 recogida del propio panel de tareas.

En apenas cuatro horas (frente a las ocho estimadas) se realizaron los últimos ajustes solicitados por el equipo de desarrollo BIM. Estas adaptaciones incluyeron la localización de cadenas de interfaz modificando etiquetas clave los elementos IFC con mayor relevancia a la hora de gestionar el modelo en el motor.

6.7.2. Tarea 7.2: Configuración multiplataforma

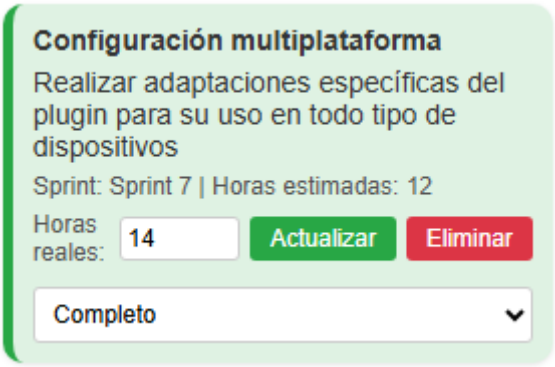

Ilustración 34: Información de la Tarea 2 del Sprint 7 recogida del propio panel de tareas.

La adaptación multiplataforma requirió 14 horas en lugar de las 12 estimadas, principalmente por la complejidad de asegurar la portabilidad de las rutas a los archivos IFC/OBJ en StreamingAssets y el retraso en las fases de pruebas (Tech Review & QA), así como la compatibilidad y rendimiento de los shaders URP en dispositivos móviles. Además, se incorporaron dos ciclos adicionales de

Tech Review & QA para validar la correcta carga de recursos en móviles. Durante este proceso se revisaron y parametrizaron los paths relativos, se implementaron validaciones de plataforma para los inputs de usuario y se añadieron fallbacks de shader cuando URP no estuviera disponible, garantizando así una experiencia fluida y consistente en todos los entornos objetivos.

6.7.3. Tarea 7.3: Documentación técnica

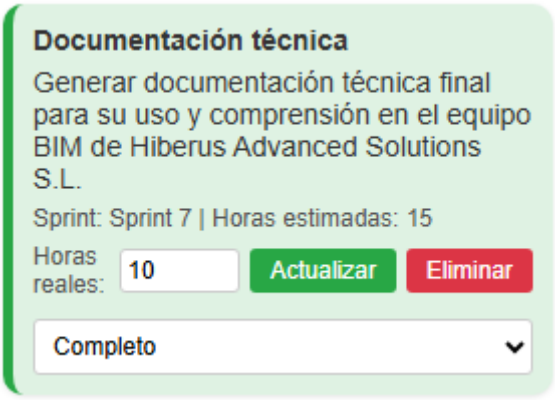

Ilustración 35: Información de la Tarea 3 del Sprint 7 recogida del propio panel de tareas.

En solo diez horas se completó la versión final de la documentación técnica en Confluence gracias al uso de plantillas y guías previas. Esta incluye un **protocolo de instalación** (versiones compatibles de Unity, xBIM, .NET e IfcOpenShell y configuración del proyecto), un **manual de equipo** con capturas e instrucciones paso a paso para importar modelos IFC, aplicar materiales y ajustar parámetros, un apartado sobre librerías open source (integración, patrones de uso y limitaciones) y **recomendaciones de buenas prácticas** para escalar el proyecto, garantizando un diseño modular que facilita la instalación, las extensiones futuras y la transferencia de conocimiento.

El desarrollo permitió implementar progresivamente un plugin BIM modular y multiplataforma: desde la investigación y elección tecnológica, el **diseño arquitectónico** y la **automatización** de conversiones y transformaciones.

El plugin importa **modelos IFC modernos** con geometría compleja, preserva su jerarquía espacial, asocia automáticamente propiedades y materiales, añade MeshColliders exactos por capa respetando el GlobalId y soporta distintas versiones IFC sin intervención manual. Funciona en entornos multiplataforma gracias a optimizaciones de memoria, fallbacks de shader y validaciones de compatibilidad. El resultado **cumple los requisitos** del equipo BIM de Hiberus Advanced Solutions S.L., ofreciendo una solución profesional y alineada con los flujos de trabajo del sector, y sienta las bases para futuras ampliaciones como edición interactiva, simulaciones de mantenimiento, integración IoT o entornos de formación inmersiva.

7. Estudio Económico

En este capítulo se realiza un análisis detallado de los **recursos financieros** para desarrollar e implementar el visor IFC en Unity, comparando los costes con las alternativas comerciales disponibles y evaluando los beneficios derivados de la adopción de una solución a medida. El estudio se divide en tres secciones: los **costes estimados del desarrollo interno**, la **comparativa con productos comerciales** (especialmente del ecosistema Autodesk y Unity Reflect) y los **beneficios potenciales** al elegir esta tecnología.

7.1. Costes estimados del desarrollo

Para estimar con rigor el coste del desarrollo, en primer lugar, se han **evaluado las horas de ingeniería** dedicadas a las distintas fases del proyecto, calculadas conforme a los importes horarios establecidos en el XIX Convenio colectivo estatal de empresas de consultoría, tecnologías de la información y estudios de mercado y de la opinión pública, registrado y publicado por la Resolución de 4 de abril de 2025 de la Dirección General de Trabajo (BOE-A-2025-7766). A continuación, se han contabilizado los gastos en licencias y herramientas de software, incluyendo tanto las soluciones comerciales (motores de desarrollo, entornos de programación y plugins específicos) como las bibliotecas open source imprescindibles para la implementación y prueba del visor BIM. Finalmente, se han incorporado los costes indirectos derivados de la infraestructura tecnológica: el mantenimiento de los servidores de desarrollo y producción, las suscripciones al ecosistema Atlassian para la gestión de proyectos y el seguimiento de incidencias, así como el espacio de almacenamiento en la nube destinado a ficheros IFC y otros artefactos generados.

7.1.1. Dedicación del Equipo de Ingeniería

El equipo de desarrollo (Ingenieros Senior especializados en BIM) distribuyó su esfuerzo a lo largo de los **siete sprints**. Las horas reales empleadas por sprint, tal y como se documentaron en cada fase, se resumen en la siguiente tabla.

Para asignar un coste por hora de ingeniería, se toma como referencia un **salario bruto medio anual** de un ingeniero de software/BIM en España de 36.000,00 €/año (fuente: convenio previamente mencionado). Suponiendo una jornada anual efectiva de 1.720 horas (contabilizando periodos vacacionales, festivos y descansos), el coste por hora se cifra en ≈ 21,00 €/h (36.000,00 € / 1.720 h). Para simplificar y redondear a valores comerciales, se utilizará 22,00 €/h como tarifa media, que incluye costes de seguridad social, infraestructura y beneficios indirectos.

Sprint	Tareas	Horas reales	Coste/hora	Coste real
1	Investigación y evaluación de oportunidades	49	22,00 €	1.078,00 €
2	Diseño arquitectónico	38,50	22,00 €	847,00 €
3	Automatización de conversión geométrica	58	22,00 €	1.276,00 €
4	Integración y visualización de metadatos	32	22,00 €	704,00 €
5	Desarrollo del parser de materiales	46	22,00 €	1.012,00 €
6	Implementación de colisiones	22	22,00 €	484,00 €
7	Adaptaciones finales	28	22,00 €	616,00 €
	TOTAL	273,50		6.017,00 €

Tabla 3: "Resumen de horas reales de trabajo y coste por sprint".

El coste total de horas de ingeniería ascendió a 6.017,00 €. Este importe incluye desde las primeras fases de investigación en octubre hasta el último ajuste de documentación a mediados de junio.

7.1.2. Costes de licencias y herramientas

Se optó por un ecosistema tecnológico basado en **herramientas gratuitas** para reducir significativamente los costes de licencias y abandonar progresivamente la dependencia del entorno cerrado de Autodesk. Para la parte gráfica y de interacción se empleó Unity Personal 6000.0.23f1, que no requiere ningún pago de licencia siempre que los ingresos del proyecto no superen los 100.000,00 USD. Al usar Unity únicamente desde el Editor (sin generar ejecutables distribuidos) no surge obligación de adquirir Unity Pro ni Enterprise a pesar de superar esos ingresos. De este modo, se dispone de un visor IFC profesional de alta calidad sin incurrir en el elevado coste anual que representan las suscripciones a software propietario.

En lugar de herramientas como Revit y Navisworks, que implican más de 5.000,00 € por usuario al año, se integró Xbim Toolkit para el procesamiento y gestión de metadatos junto a IfcOpenShell para las distintas conversiones; ambas librerías son de código abierto y no generan ningún coste de licencia. El entorno de desarrollo se completó con Visual Studio Community 2022, gratuito para proyectos de esta magnitud, y la documentación se centralizó en Atlassian Confluence, aprovechando la suscripción corporativa existente (aprox. 60,00 € mensuales) sin necesidad de abonar licencias adicionales. Incluyendo el sistema de tareas alojado en el propio servidor web con un coste inicial de 14,80 € (adquisición de dominio) y un coste posterior de 1,80 €/mes (29,20 € totales).

Esta estrategia no solo **minimiza drásticamente la inversión** en licencias y servicios, sino que también **evita la dependencia** de plataformas propietarias, proporcionando al cliente una solución más económica, flexible y alineada con estándares abiertos.

7.1.3. Costes de hardware

El ingeniero principal desarrolló el proyecto sobre un equipo portátil de altas prestaciones (GIGABYTE AORUS 15P XD-73ES324SH) adquirido previamente cuyas especificaciones dan a entender la plataforma mínima requerida y el coste real de uso durante el periodo de octubre a junio.

CPU	Intel Core i7-11800H
GPU	NVIDIA GeForce RTX 3070 Laptop
RAM	16 GB DDR4 3200 MHz
Almacenamiento	1 TB SSD NVMe
Pantalla	15,6” FHD IPS 300 Hz

Tabla 4: “Costes e información de hardware”.

El precio de compra del equipo fue de 1.098,00 € (I.V.A. incluido) en la web PCComponentes, con una garantía de 2 años y una vida útil estimada de 5 años de uso (1.826 días). De este modo, el coste de amortización del portátil se sitúa en aproximadamente 0,025 €/h. Si añadimos además un consumo energético de 180 W (0,18 kW) del dispositivo a un rendimiento medio-alto, y considerando que el precio medio del kWh en España actualmente es de 0,1463 €/kWh (fuente: tarifaluzhora.es a finales de mayo de 2025), el gasto energético por hora de uso es:

- Consumo por hora (kWh): 0,18 kW × 1 h = 0,18 kWh
- Coste energético por hora: 0,18 kWh × 0,1463 €/kWh ≈ 0,0263 €/h

Por tanto, el coste total por hora (amortización + energía) asciende a:

- 0,025 €/h + 0,0263 €/h = 0,0513 €/h.

En consecuencia, cada hora de trabajo con el portátil implica un gasto aproximado de 0,0513 €, lo que permite dimensionar con mayor precisión los costes operativos del desarrollo para las 273,50 horas de desarrollo que sitúan un coste final en hardware de 14,03€.

El coste unificado asciende a 6.017,00€ (ingeniería) + 14,03€ (hardware/servicios) + 89,20€ (licencias) = 6.120,03€.

7.2. Comparativa de costes con soluciones comerciales

A la hora de evaluar el impacto económico de desarrollar internamente el visor IFC en Unity, resulta conveniente **contrastar estos gastos** con los asociados a la adquisición y mantenimiento de software BIM comercial. En esta sección se presenta un resumen simplificado de los costes anuales por usuario de las principales herramientas empleadas habitualmente en los flujos BIM, seguido de un análisis comparativo con la solución interna.

7.2.1. Principales licencias BIM y sus costes

La *Tabla 1* de la *sección 2.2* recoge, de forma concisa, los programas BIM más utilizados en el sector ligado a costes, las áreas de aplicación básicas y su coste aproximado de suscripción anual por usuario.

En un equipo BIM típico, **los profesionales combinan varias licencias** de software del ecostistema de Autodesk para cubrir todo el flujo de trabajo: diseño arquitectónico en Revit, coordinación de disciplinas y detección de interferencias en Navisworks Manage, renderizado en 3ds Max o Rhino/VisualARQ, modelado de ingeniería civil con Civil 3D y diseño estructural con Tekla Structures. Solo el uso conjunto de Revit y Navisworks Manage, dos de las herramientas básicas para modelar y coordinar gigantescos proyectos de construcción, supone un coste aproximado de 3.338,00 € por usuario al año para Revit y 3.195,00 € por usuario al año para Navisworks. Por tanto, un equipo de cinco profesionales BIM que requiriera ambas licencias **desembolsaría alrededor de 32.665,00 € anuales** (es decir, 5 × 3.338,00 € + 5 × 3.195,00 €) solo en suscripciones de esos dos productos.

En el caso particular de Grupo LOBE, actualmente se mantienen suscripciones tanto de Revit como de Navisworks Manage para dos usuarios simultáneos cada una. Esto implica un gasto

anual de 6.676,00 € por las dos licencias de Revit (2 × 3.338,00 €) y 6.390,00 € por las dos licencias de Navisworks Manage (2 × 3.195,00 €), situando el coste combinado de ambas herramientas en 13.066,00 € al año.

La solución interna desarrollada en este proyecto **permite conservar únicamente las dos licencias de Revit**, eliminando por completo la necesidad de Navisworks Manage. Gracias al visor IFC implementado en Unity para la conversión, todas las tareas de coordinación, detección de interferencias y simulación de escenas que antes se realizaban en Navisworks se podrán manejar de forma local dentro de Unity. De este modo, el único desembolso de software propietario que permanece es el de Revit, con un coste de 6.676,00 € anuales para dos licencias, ahorrando así los 6.390,00 € que hasta ahora destinaban a Navisworks Manage.

Esta estrategia no solo **reduce drásticamente la inversión en suscripciones**, sino que también **mantiene intacto el flujo de trabajo** en Revit evitando la curva de aprendizaje y los costes de migración que supondría reemplazar por completo la plataforma de Autodesk. En definitiva, el proyecto ofrece la capacidad de continuar utilizando Revit para el modelado paramétrico y el dibujo, a la vez que dispone de funcionalidades equivalentes o superiores a las de Navisworks para la coordinación y visualización, pero sin afrontar las tasas anuales que conlleva dicha herramienta propietaria.

7.3. Beneficios potenciales de la implementación

La adopción de la solución interna basada en Unity y tecnologías de código abierto genera un ahorro económico inmediato y sostenido. En el caso de Grupo LOBE, prescindir de las dos licencias de Navisworks Manage (6.390,00 € al año) y mantener únicamente las dos licencias de Revit (6.676,00 € anuales) implica una reducción directa del presupuesto en 6.390,00 €. Este importe equivale prácticamente al coste total de desarrollo y hardware (6.120,03 €) recuperado en el primer año. A partir del segundo año, solo se incurriría en un coste menor por mantenimiento del software interno (aproximadamente el 10 % del esfuerzo de desarrollo original, es decir, 601,70 € anuales), lo que permite liberar recursos para otras iniciativas.

Otro beneficio radica en la **eliminación de costes** recurrentes asociados a plataformas de visualización y coordinación comerciales. Herramientas como Unity Reflect pueden exigir licencias anuales por usuario de 3.100,00 € sin incluir posibles servicios en la nube. La implementación interna no genera pagos adicionales de licencia. Tras la inversión inicial de 6.017,00 € en horas de ingeniería, 14,03 € en amortización de hardware y 89,20 € en suscripciones corporativas de Confluence, el proyecto continúa funcionando sin tarifas anuales. Solo se necesitarían horas puntuales de actualización y soporte (estimadas en 601,70 € al año), lo que sitúa el coste anual de operación muy por debajo de las alternativas propietarias.

Además, se **ahorra en formación y migración**. El equipo BIM permanece trabajando en Revit, eliminando la necesidad de invertir tiempo y dinero en cursos de Navisworks o Unity Reflect. La única formación adicional requerida para utilizar el visor IFC en Unity se limita a una breve sesión introductoria (1,0–2,0 h), lo cual representa un coste mínimo en horas de trabajo y evita el pago de cuotas de capacitación externa. Esta reducción de la curva de aprendizaje se traduce en una mejora de la productividad desde el primer mes de uso.

A medio y largo plazo, el desarrollo propio ofrece un coste de propiedad mucho más bajo. En escenarios típicos, un equipo de cinco usuarios BIM que utilice Revit y Navisworks sufrirá 32.665,00 € anuales en licencias. Si dichos usuarios migran a la solución interna, con Unity Personal y herramientas open source, el único coste periódico correspondería a las actualizaciones internas (601,70 € al año) y, opcionalmente, a la mejora de hardware o licencias adicionales de Revit. Esto **libera más de 30.000,00 € cada año**, recursos que pueden destinarse a inversión en nuevos desarrollos, adquisición de equipos de visualización avanzados o servicios de consultoría.

Finalmente, aunque el principal objetivo fue optimizar costes, **existe un potencial de ingresos adicionales**. Si se comercializara el visor como producto o se ofrecieran servicios de personalización, la inversión inicial de 6.120,03 € podría recuperarse en múltiples proyectos, convirtiéndose en una fuente de ingresos. Esto incrementaría el retorno de inversión y posicionaría a la organización como referente en soluciones BIM personalizadas.

7.4. Resumen económico

El coste total de desarrollar internamente el visor BIM en Unity se ha estimado en torno a **6.120,00 €,** abarcando principalmente horas de ingeniería y, en menor medida, gastos en licencias y hardware. Gracias a la adopción de herramientas **open source** y versiones gratuitas (como Unity Personal), los costes directos de licencias resultaron prácticamente **nulos**, y la inversión en equipamiento se limitó a la amortización del hardware existente.

Al comparar este desarrollo propio con las **soluciones comerciales** del mercado, la **ventaja económica** se hace evidente. Herramientas BIM tradicionales, como Autodesk Navisworks, suponen costes de licenciamiento sustanciales (varios miles de euros por usuario y año). En nuestro caso de estudio, eliminar Navisworks y optar por el plugin desarrollado libera aproximadamente **6.400,00 € anuales** (considerando dos licencias), cifra que **iguala e incluso supera el coste total del desarrollo interno en solo un año**. Del mismo modo, evitar el uso de plataformas como Unity Reflect (con cuotas en torno a 3.100,00 € por usuario/año) implica prescindir de pagos recurrentes significativos. Es decir, la solución desarrollada no solo **evita**

costes anuales elevados, sino que permite **recuperar la inversión inicial en el corto plazo** mediante los ahorros conseguidos.

La adopción de una plataforma propia basada en tecnologías abiertas conlleva además **beneficios directos** más allá del ahorro inmediato. En primer lugar, se elimina la **dependencia de proveedores externos** y de formatos propietarios. La organización obtiene plena autonomía para adaptar y mejorar la herramienta según sus necesidades, sin estar sujeta a cambios de licencias o discontinuidades de servicio (como la reciente retirada anunciada de Navisworks). En segundo lugar, se reducen los **costes ocultos** asociados a la transición y formación: al conservar los flujos de trabajo en software ya conocido (p. ej., Revit para el modelado) e integrar el visor IFC de forma transparente, el equipo minimiza la curva de aprendizaje y **evita gastos en capacitación** para nuevas herramientas comerciales.

En términos de **retorno de la inversión (ROI)** y sostenibilidad a medio-largo plazo, los resultados son muy favorables. La inversión inicial queda **amortizada en aproximadamente un año**, dado que el ahorro anual en licencias propietarias iguala el coste de desarrollo del plugin. A partir del segundo año, el proyecto entra en **retorno positivo**. Los únicos gastos esperados son los de mantenimiento y mejoras incrementales, estimados en una fracción pequeña del esfuerzo original (por ejemplo, alrededor del 10 % anual del desarrollo, unos **600,00 €**). Esto se traduce en un **coste de propiedad** significativamente inferior al de las alternativas comerciales: cada ejercicio adicional de uso de la solución propia representa ahorros netos acumulativos, reforzando su **viabilidad financiera a largo plazo**.

Cabe destacar, por último, que este desarrollo no solo ha demostrado su eficacia técnica y viabilidad económica, sino que ha suscitado **interés comercial real**. Actualmente, se encuentra en proceso de **comercialización con FCC** (Fomento de Construcciones y Contratas), con un primer contrato valorado en **12.900,00 € (relativo a una prueba de concepto PoC)**. Esta operación no solo valida externamente el valor del producto, sino que anticipa su potencial de **escalabilidad en el sector de la construcción**. Empresas del ámbito AEC con flujos de trabajo basados en BIM, y especialmente aquellas que requieren soluciones multiplataforma, ligeras y sin dependencia de licencias privativas, serán candidatas naturales para **integrar este visor IFC** en sus entornos productivos. Todo ello refuerza la proyección del proyecto no solo como una solución técnica eficaz, sino también como un **activo comercial competitivo** con recorrido en el mercado profesional.

8. Resultados

En este capítulo se presenta la evaluación integral del plugin, midiendo su grado de cumplimiento frente a los requisitos funcionales, de rendimiento y de estabilidad definidos. La validación abarca tanto el correcto funcionamiento de cada módulo (carga de geometría, extracción de metadatos, asignación de materiales y generación de colisiones) como la comparación con soluciones comerciales en términos de usabilidad, automatización y adaptabilidad. Para ello se han ejecutado pruebas controladas en entornos de escritorio y dispositivos móviles con modelos IFC de distinta complejidad, con el fin de demostrar que el visor IFC en Unity satisface las necesidades reales de visualización, consulta y análisis de datos BIM, ofreciendo una experiencia fluida y una arquitectura extensible. A continuación, se detalla el análisis de la evaluación funcional del plugin.

8.1. Evaluación del plugin desarrollado

La evaluación del plugin se ha articulado en dos niveles principales: primero, se **verifica su correcto funcionamiento** bajo las condiciones de uso previstas; segundo, se analiza su **capacidad** para importar, vincular y mostrar modelos IFC de diversa complejidad, identificando los posibles errores o puntos de mejora. Para ello, se establecieron una serie de criterios de validación funcional que comprenden desde la interfaz de usuario hasta la sincronización de datos y la robustez ante archivos IFC con variantes o extensiones propietarias.

8.1.1. Interfaz de carga y gestión de archivos IFC

Para facilitar el proceso de importación, el plugin incorpora una **ventana de Editor** (*EditorWindow*) que permite introducir una URL remota o seleccionar un fichero local. En la imagen siguiente se muestra el panel "Cargar IFC desde URL", en el que el usuario puede introducir la dirección web de un modelo IFC y descargarlo directamente al proyecto de Unity. En el presente caso se ha introducido el edificio sobrecargado mencionado anteriormente, el acceso a él está restringido mediante la conexión VPN del cliente y equipo.

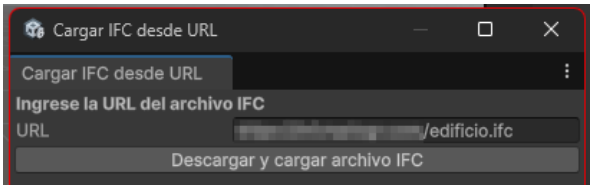

Ilustración 36: "Ventana de carga de IFC desde URL", donde el usuario introduce la ruta y descarga y carga el fichero automáticamente.

El mecanismo asíncrono de descarga y conversión se activa al pulsar el botón "Descargar y cargar archivo IFC", lo que desencadena el proceso *IfCFileLoader.LoadFromUrl()*. Durante esta fase, el

plugin muestra un **indicador de progreso** en la consola informa en tiempo real del escaneo y la generación de mallas. La captura de pantalla que sigue ilustra cómo se visualiza en la terminal el avance de IfcOpenShell al procesar un modelo de tamaño elevado.

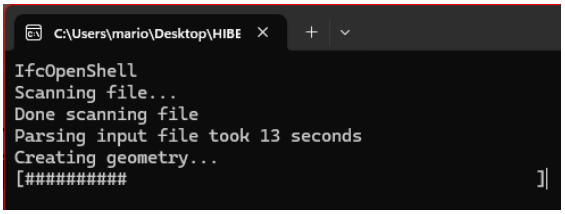

Ilustración 37: "Consola de carga de modelos".

Se verificó que, en un equipo equipado con un procesador Intel i7-11800H y una GPU RTX 3070 Laptop, el plugin es capaz de procesar un modelo IFC de 161 MB (aproximadamente 3 millones de líneas de código) en **apenas trece segundos**, tiempo necesario para el parseo y la validación inicial de datos. A continuación, la fase de generación geométrica, que abarca la creación de mallas para los **7.338 objetos** del modelo, equivalentes a cerca de **7 millones de triángulos**, requiere aproximadamente nueve minutos. Finalmente, la asignación de todos los metadatos asociados a cada objeto se completa en torno a un minuto adicional.

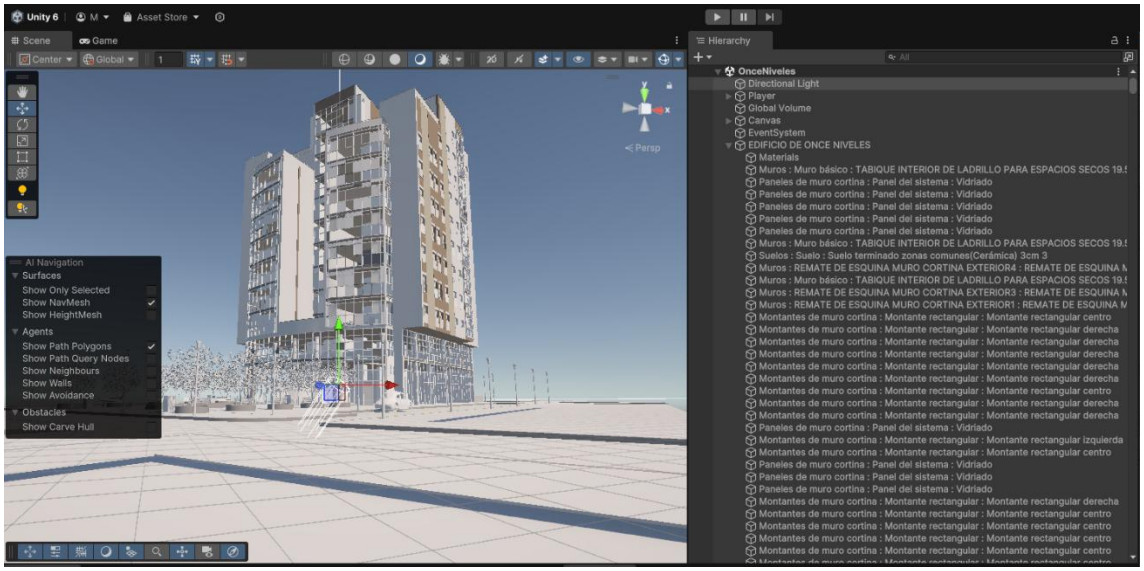

Ilustración 38: "Modelo completamente cargado desde el Editor". Captura tomada desde el Editor en Unity 6 una vez cargado completamente el modelo.

En consecuencia, los tiempos de carga para modelos de pequeño tamaño (menores a 5 MB) resultan **prácticamente instantáneos**, mientras que aquellos proyectos de gran complejidad geométrica, si bien tardan varios minutos en construir su malla, se importan con una velocidad notablemente rápida teniendo en cuenta la escala de datos procesados. Esta combinación de

tiempos de parseo, generación de geometría y asignación de metadatos asegura un flujo de trabajo eficiente, incluso en casos de modelos pesados, manteniendo la experiencia de usuario **ágil y sin bloqueos** prolongados en el Editor.

8.1.2. Visualización final del modelo

A pesar del enorme volumen de geometría, el visor es capaz de mantener un rendimiento óptimo. Con parámetros altos, la escena completa, es decir, todo el modelo cargado dentro del *frustrum* de la cámara, se mantiene por **encima de 60 FPS**. Este rendimiento permite al usuario recorrer libremente el entorno sin experimentar caídas significativas en la tasa de fotogramas, incluso con **detalles complejos** como mobiliario y vegetación incluidos. Una vez optimizado durante el desarrollo utilizando la herramienta que ofrece Unity del profiler en los siguientes resultados se utiliza la superposición de NVIDIA que permite mostrar el rendimiento en tiempo real en la parte superior derecha de la build generada.

Ilustración 39: "Edificio de Once Niveles importado". Captura de la build creada (escritorio).

Ilustración 40: "Ejemplo de rendering comercial". Muestra de rendering comercial del mismo edificio importado previamente.

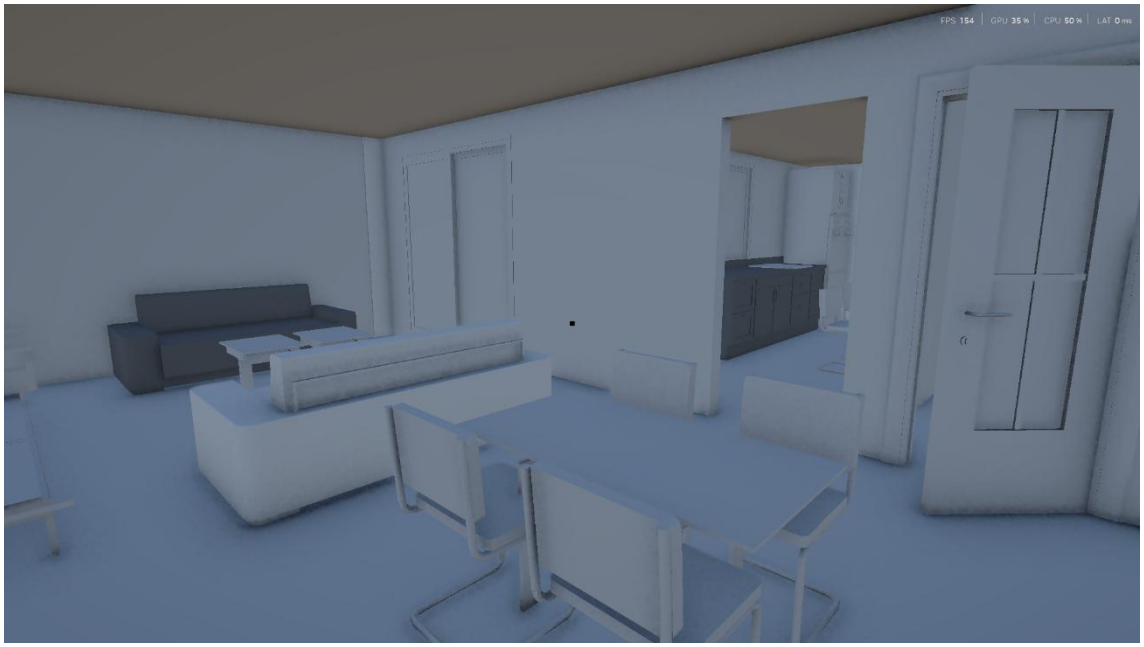

Ilustración 41: "Interior del modelo". Captura de la build creada.

Sin embargo, el rendimiento en **interiores crece enormemente** con más de 160 frames por segundo como se puede apreciar en la ilustración anterior o incluso más de 250 como se apreciará en siguientes ilustraciones visualizando la parte superior derecha de la misma.

8.1.3. Gestión de materiales

El modelo cuenta con unos 120 materiales distintos, donde prima la **riqueza informativa** sobre la perfección visual, de modo que algunas texturas aún no alcanzan el nivel de un render comercial final. La asignación en Unity es prácticamente instantánea, ya que el sistema lee las etiquetas *SurfaceStyleLabel* del archivo MTL y genera en tiempo real los *assets* Material. Además, el botón "Aplicar materiales" permite crear gráficos personalizados para cada acabado y ajustar parámetros como albedoColor, metallic y smoothness o aplicar texturas propias según las necesidades. Por ejemplo, las superficies transparentes (paneles de vidrio) se renderizan correctamente activando alphaClipping y el modo transparente en URP, logrando reflejos y refracciones simples sin ajustes manuales adicionales.

Ilustración 42: "Materiales del modelo". Listado parcial de los materiales importados, donde se aprecia la correcta generación de materiales opacos y transparentes.

Gracias a esta **flexibilidad**, cada vez que se incorporan variantes de BIM con diferentes acabados según los requerimientos: únicamente visor o renderización comercial, basta con actualizar o crear los materiales y pulsar "Aplicar materiales" para que el visor asigne de inmediato toda la paleta de texturas, sin interrupciones ni tiempos de esper. Esto facilita la experimentación con **distintas opciones visuales** (por ejemplo, acabados cerámicos, metales pulidos o vidrios tintados) sin alterar la integridad de los metadatos, lo que resulta en un equilibrio óptimo entre precisión informativa y calidad gráfica.

8.1.4. Gestión de metadatos

Para aprovechar al máximo la riqueza informativa del modelo BIM, el plugin ofrece **dos modos complementarios de visualización de metadatos**: en el Editor de Unity y en las builds generadas.

En el Editor, basta con seleccionar cualquier objeto en la jerarquía o en la escena para que el Inspector muestre automáticamente toda su información asociada. A medida que el usuario expande cada *PropertySet*, podrá consultar los valores de cada propiedad (por ejemplo, "*LoadBearing*", "*FireRating*" o "*RatedHeight*") sin necesidad de pasos adicionales. Esta visualización en el Inspector resulta inmediata y permite desplazarse por los distintos conjuntos de propiedades utilizando la rueda del ratón o las flechas de teclado.

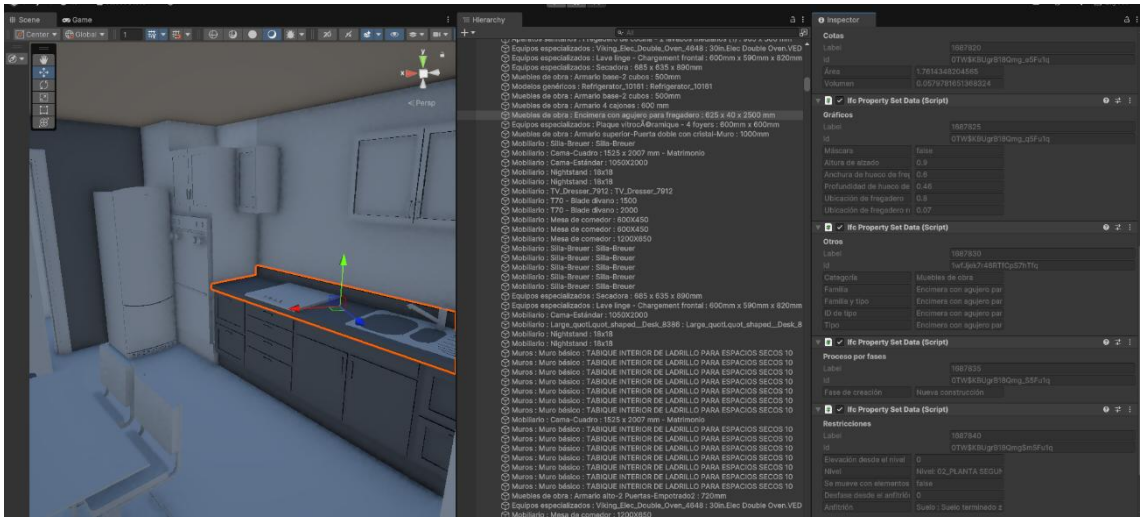

Ilustración 43: "Información de elemento en el Editor". Captura tomada desde el Editor de Unity 6.

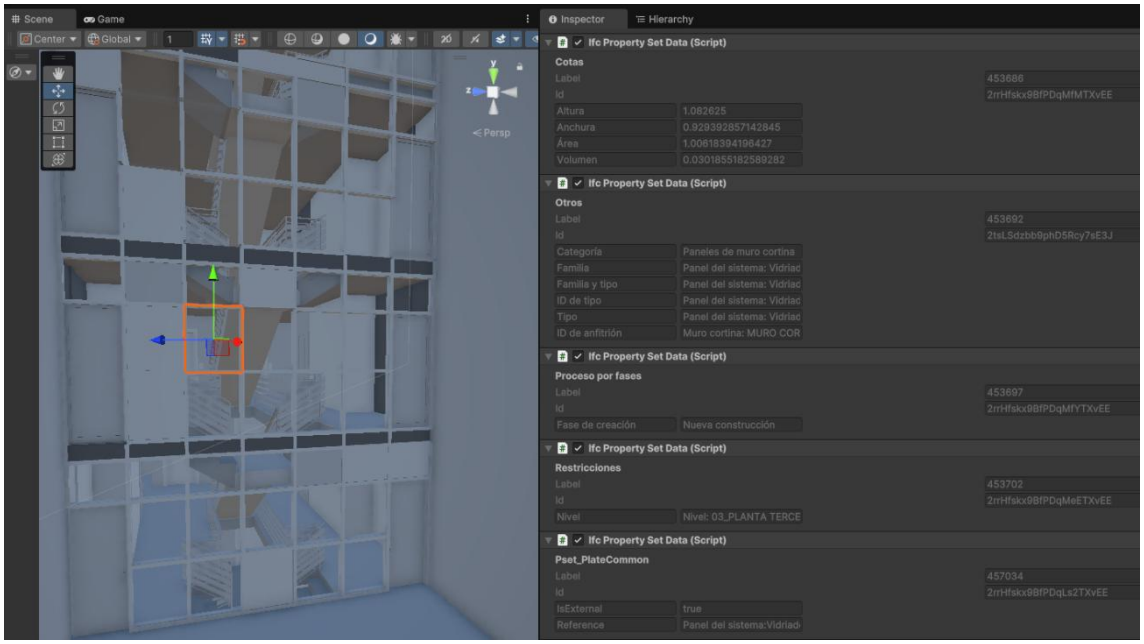

Ilustración 44: "Información y visualización de un vidrio exterior". Captura tomada desde Unity 6.

En una build ejecutable, la interacción se adapta a entornos 3D en tiempo real: al **apuntar con el retículo central hacia un objeto**, este se resalta con un material de selección (amarillo brillante). A continuación, el usuario debe pulsar la tecla o el botón designado según la plataforma de ejecución para abrir el **panel flotante de metadatos**. Una vez abierto, las propiedades se presentan en un panel superpuesto en la esquina superior izquierda de la pantalla; desde ahí, el usuario puede desplazarse hacia abajo o hacia arriba utilizando las teclas de flecha, el joystick o la rueda del ratón, según el dispositivo pudiendo filtrar por contenidos. Asimismo, para facilitar la navegación dentro del modelo, se ha incorporado un mecanismo que al apuntar con el retículo central a una puerta y activar la tecla correspondiente, desactiva temporalmente la puerta, permitiendo así el **desplazamiento sin obstáculos**.

Ilustración 45: "Visualización de un tabique en zona de estar". Captura tomada desde la propia build pudiendo interactuar con sus propiedades.

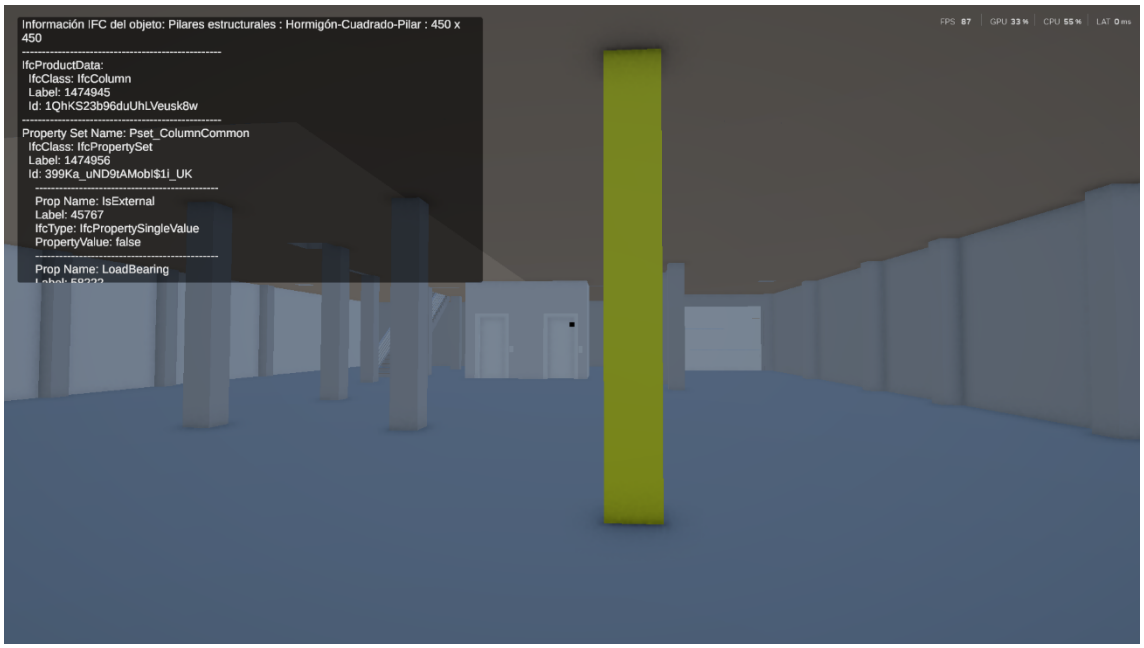

Ilustración 46: "Visualización de un pilar del garage en la build". Captura tomada desde la propia build con el pilar seleccionado pudiendo interactuar con las propiedades.

8.2. Comparación con herramientas comerciales

Las plataformas de Autodesk (Revit para modelado BIM y Navisworks Manage para coordinación 3D/4D) ofrecen un flujo completo, pero suponen un **gasto anual muy elevado** en licencias y actualizaciones. Además, su personalización en la visualización 3D está restringida por APIs parciales y, para ajustar asignación de materiales o colisiones, es necesario recurrir a extensiones de pago o **servicios externos**. Por su parte, Unity Reflect (Enterprise) sincroniza en tiempo real con Revit y aporta soporte VR/AR, pero exige suscripción a Unity Pro y depende de servicios en la nube, lo que encarece el coste de propiedad y obliga a desarrollos adicionales para gestionar metadatos IFC o colliders optimizados.

En cambio, el visor IFC interno funciona con Unity Personal y **librerías open source**, sin incurrir en nuevas licencias comerciales para la parte gráfica. Esta independencia permite reducir el coste total de propiedad a dos años y ofrece plena libertad para ajustar la tessellation, reconfigurar módulos físicos o exportar datos, sin esperar a *roadmaps* externos. Además, cualquier error en la interpretación de un IFC o en la aplicación de materiales se corrige en horas, en lugar de semanas, y la misma herramienta cubre Windows, macOS, Android, iOS y WebGL sin costes extra de plataforma. En definitiva, más allá del ahorro económico, la solución propia proporciona un nivel de **flexibilidad y personalización** que las opciones comerciales estándar no alcanzan.

8.3. Aportaciones y valor diferencial del proyecto

La propuesta de un visor IFC propio aporta varias ventajas clave que lo distinguen de las alternativas cerradas y refuerzan su potencial a largo plazo:

1. **Solución propia y alto grado de escalabilidad**: Al disponer del código fuente completo, el equipo puede integrar rápidamente nuevas versiones de IFC (por ejemplo, Ifc5 o extensiones sectoriales futuras) y añadir **módulos de simulación a medida** (análisis energético, simulaciones estructurales o integración con IoT) sin depender de actualizaciones de terceros. Este nivel de autonomía convierte la herramienta en un núcleo escalable para cualquier proyecto de digital twin, donde BIM, datos de sensores y lógica de negocio convivan unificados.
2. **Digital twins y simulaciones personalizadas**: Gracias a la extracción y vinculación nativa de metadatos, resulta trivial incorporar **variables externas** (por ejemplo, datos de ocupación o condiciones climáticas) a las entidades IFC para generar gemelos digitales que se actualizan en tiempo real. De la misma manera, el usuario puede crear simulaciones personalizadas como recorridos virtuales con *triggers* para activar eventos o integraciones con sistemas de control de accesos sin necesidad de desarrollar un nuevo motor desde cero.
3. **Abaratamiento de costes operativos y de formación**: Elimina la inversión en licencias anuales para visualización y coordinación, deja intacto el flujo de trabajo en Revit y suprime la curva de aprendizaje en nuevas plataformas. Este factor agiliza la incorporación de nuevos técnicos y libera presupuesto para otras iniciativas de I+D o hardware especializado.
4. **Modularidad y personalización total**: Cada componente se ha diseñado para funcionar de forma independiente. Esto abre la puerta a flujos **gráficos avanzados**, como la sustitución automática de shaders URP por versiones optimizadas para VR o a la creación de efectos personalizados según requerimientos de cliente.
5. **Interoperabilidad y alineamiento con estándares abiertos**: Al usar xBIM Toolkit e IfcOpenShell, la solución refleja las **últimas novedades** de la comunidad IFC. Esto garantiza que el visor no quede obsoleto.

El proyecto aporta un valor diferencial que fusiona las ventajas de un **sistema propietario** (control total, rendimiento optimizado, adaptabilidad) con el ahorro de costes e independencia de un ecosistema open source. Estas características sitúan a la solución en una posición única para cualquier empresa que quiera incorporar visualización BIM de alta calidad, gemelos digitales y simulaciones a medida, sin sacrificar la autonomía ni asumir gastos recurrentes elevados.

9. Conclusiones y Futuras Ampliaciones

En este capítulo se sintetizan los resultados más relevantes del proyecto, se señalan las limitaciones detectadas y se proponen mejoras para ampliar la herramienta propuesta en este TFG en el futuro.

9.1. Conclusiones generales

El desarrollo de un visor IFC basado en Unity y tecnologías de código abierto ha demostrado ser **viable y rentable**. Utilizando Unity Personal y las librerías *xBIM Toolkit* e *IfcOpenShell*, se ha construido una solución **capaz de importar modelos BIM completos**, mostrar metadatos, asignar materiales y generar colisiones de manera automática, sin incurrir en licencias comerciales adicionales para la parte gráfica. Los tiempos de carga y procesamiento resultan aceptables incluso para modelos muy complejos (161 MB, 7.338 objetos, ≈ 7 millones de triángulos), con una fase de parseo de datos en 13 segundos, construcción de la geometría en nueve minutos y asignación de metadatos en un minuto. En escritorio, la renderización se mantiene por encima de 60 FPS con tessellation alta renderizando todo el modelo.

Gracias a la extracción completa de metadatos del modelo IFC, el usuario puede consultar cualquier **atributo en tiempo real**, ya sea en el Editor o en la versión ejecutable. El sistema de materiales URP distribuye todos los acabados en instantes y permite personalizaciones dinámicas, mientras que los *colliders* se asignan automáticamente según la complejidad de cada entidad basándose en su propia malla, garantizando un equilibrio adecuado entre precisión física y rendimiento. Todo esto se consigue incluso reduciendo los costes por lo que es una vía que hay que trabajar.

9.2. Limitaciones del proyecto

A pesar de sus buenos resultados, el visor BIM propuesto en este TFG presenta ciertas limitaciones. Primero, el visor depende de **herramientas open source externas**, lo cual impone restricciones en plataformas que no admiten binarios nativos. Segundo, la gestión de memoria con modelos muy grandes puede superar la capacidad de **dispositivos de gama baja**, por lo que sería necesario implementar técnicas de *streaming* o niveles de detalle dinámico. Tercero, el **formato IFC sigue evolucionando constantemente**; ello obliga a mantener actualizados los *parsers* y validadores para que nuevas versiones (Ifc5.x, extensiones sectoriales o conjuntos de propiedades personalizados) se importen, muestren y validen sin errores en el visor. En cualquier caso, hay que tener en cuenta que el ecosistema BIM en general es todavía muy dinámico y sujeto a cambios frecuentes.

9.3. Propuestas de mejora y futuras líneas de trabajo

Varias mejoras permitirían no solo superar las limitaciones actuales, sino también ampliar significativamente las capacidades de la solución:

1. **Desarrollo de un motor propio**: Crear un motor gráfico y de importación BIM completamente propio permitiría eliminar la dependencia de Unity y sus restricciones de licencia.
2. **Streaming de malla y LOD dinámico**: Implementar un sistema que cargue progresivamente la geometría según la distancia a la cámara aliviaría el consumo de memoria en dispositivos de recursos limitados.
3. **Shaders avanzados y materiales PBR**: Integrar efectos de normal mapping, mapas de rugosidad y oclusión ambiental mejoraría la calidad visual, acercándose a estándares de renderizado profesional.
4. **Animaciones de componentes BIM**: Añadir controladores personalizados que habiliten la animación de puertas, ventanas y equipos movería el visor desde una herramienta pasiva a un entorno interactivo. Las animaciones se basarían en parámetros IFC (ángulo de apertura, recorridos de elementos móviles) y permitirían simular flujos de usuarios o ciclos operativos sin depender de software adicional.
5. **Gemelos digitales y datos en tiempo real**: Exponer una API REST/WebSocket para conectar el visor con sensores IoT y sistemas de gestión de edificios introduciría capacidad de monitorización en vivo. De este modo, valores como temperatura, humedad o estado de ocupación podrían vincularse a entidades IFC específicas, transformando el visor en una plataforma de gemelos digitales que refleja cambios del entorno físico al instante.
6. **Interfaz modular y personalizable**: Diseñar un sistema de menús y paneles flotantes que el usuario pueda activar o desactivar según su función (técnico, comercial, cliente final) facilitaría la experiencia de usuario.

Implementar estas mejoras consolidaría la herramienta como un estándar abierto en visualización BIM, a la vez que habilitaría nuevas aplicaciones en simulaciones personalizadas, mantenimiento a través de gemelos digitales e integración constante con flujos de datos en tiempo real. De este modo, el visor **evolucionaría hacia una plataforma integral** para proyectos de arquitectura, ingeniería y construcción, capaz de adaptarse con agilidad a las demandas futuras del sector.

10. Bibliografía

[1] AZHAR, Salman. Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry. Leadership and management in engineering, 2011, vol. 11, no 3, p. 241-252. Disponible en: https://doi.org/10.1061/(ASCE)LM.1943-5630.0000127

[2] HYARAT, Esraa; HYARAT, Tasneem; AL KUISI, Mustafa. Barriers to the implementation of building information modeling among Jordanian AEC companies. Buildings, 2022, vol. 12, no 2, p. 150. Disponible en: https://doi.org/10.3390/buildings12020150

[3] C. Sun, X. Zhang, M. J. Skibniewski, S. Wan, L. Shen, and L. Xia, "A literature review of the factors limiting the application of BIM in the construction industry," Technological and Economic Development of Economy, vol. 23, no. 5, pp. 764–779, 2017. Disponible en: https://doi.org/10.3846/20294913.2015.1087071

[4] BRYDE, David; BROQUETAS, Martí; VOLM, Jürgen Marc. The project benefits of building information modelling (BIM). International journal of project management, 2013, vol. 31, no 7, p. 971-980. Disponible en: https://doi.org/10.1016/j.ijproman.2012.12.001

[5] CUPA Stone, "Current state of BIM in the major countries of the world," Blog CUPA Stone, 02/2018. Disponible en: https://www.cupastone.com/bim-countries-world/

[6] Autodesk, "Caso de Éxito Grupo LOBE – BIM+HUBE: Tecnología inteligente para la virtualización y optimización de los procesos industrializados en el sector de la construcción," Autodesk Campaigns - EMEA, 2017. Disponible en: https://damassets.autodesk.net/content/dam/autodesk/www/campaigns/emea/docs/lobe-casodeexito-final.pdf

[7] G. Van Berlo, "Open Source IFC Frameworks: some experiences," CAD, BIM & 3D Blog, 25 Ago 2011. Disponible en: http://cad-3d.blogspot.com/2011/08/open-source-ifc-frameworks-some.html

[8] MOSLER, Pascal; STEITZ, Nicholas-Andre Edgar. Towards a bidirectional real time link between BIM software and the game engine unity. Ruhr-Universität Bochum, 2023. Disponible en: https://www.researchgate.net/profile/Pascal-Mosler/publication/374249714_Towards_a_Bidirectional_Real_Time_Link_between_BIM_Software_and_the_Game_Engine_Unity/links/651576682c6cfe2cc2156a3f/Towards-a-Bidirectional-Real-Time-Link-between-BIM-Software-and-the-Game-Engine-Unity.pdf

[9] Autodesk, "Autodesk BIM 360 Team – End of Life Notice," Autodesk Knowledge Network, Dec. 2022. Disponible en: https://www.autodesk.com/es/support/technical/article/caas/tsarticles/tsarticles/ESP/ts/1upmroGRB3eMUWNbnb3pt9.html

[10] xBIM Team, "xBIM Toolkit – Documentation,". Disponible en: https://docs.xbim.net

[11] IfcOpenShell, "IfcOpenShell – The open source IFC toolkit and geometry engine," Disponible en: https://ifcopenshell.org

[12] BIM Corner. IfcRelationship in Infrastructure (IFC 4.3) [en línea]. Sin fecha. Disponible en: https://bimcorner.com/ifcrelationship-in-infrastructure-ifc-4-3/#:~:text=,ifcRelNests

[13] buildingSMART. IfcRelAssociatesMaterial [en línea]. Sin fecha. Disponible en: https://ifc43-docs.standards.buildingsmart.org/IFC/RELEASE/IFC4x3/HTML/lexical/IfcRelAssociatesMaterial.htm

[14] MORSE, Christopher. Gaming engines: Unity, unreal, and interactive 3D spaces. 2021. Disponible en: https://doi.org/10.1080/24751448.2021.1967068

[15] CAD-3D.BLOGSPOT. Getting BIM data into Unity – Part 8. 2018. Disponible en: https://cad-3d.blogspot.com/2018/08/getting-bim-data-into-unity-part-8.html

[16] XRBOOTCAMP. Unity vs Unreal Engine for XR Development. Disponible en: https://xrbootcamp.com/unity-vs-unreal-engine-for-xr-development/#:~:text=C,maintenance%20when%20the%20app%20grows

[17] ZAMAN, Ahmad Akib Uz; ABDELATY, Ahmed; SOBUZ, Md Habibur Rahman. Integration of BIM data and real-time game engine applications: Case studies in construction safety management. J. Inf. Technol. Constr., 2024, vol. 29, p. 117-140. Disponible en: https://doi.org/10.36680/j.itcon.2024.007

[18] AEC MAG. Unity Reflect Review and Unity Reflect Develop Launch. Disponible en: https://aecmag.com/vr-mr/unity-reflect-review-and-unity-reflect-develop-launch-revit-rhino-bim360-sketchup/#:~:text=With%20the%20launch%20of%20Unity,although%20this%20is%20in%20development

[19] BUILDINGSMART Technical. Standards / IFC. Disponible en: https://technical.buildingsmart.org/standards/ifc/#:~:text=In%20general%2C%20IFC%2C%20or%20%E2%80%9CIndustry,its%20goal%20to%20promote%20openBIM%C2%AE

[20] INTERSCALE EDU. BIM Software Limitations: Technical, Performance, and Cost-Related Challenges. Disponible en: https://interscaleedu.com/en/blog/bim/bim-software-limitations/#:~:text=,and%20interoperability

[21] WIKIPEDIA. Industry Foundation Classes. Disponible en: https://en.wikipedia.org/wiki/Industry_Foundation_Classes#:~:text=Because%20of%20its%20focus%20on,for%20the%20purpose%20of%20exchanging

[22] SALZANO, Antonio, et al. Systematic literature review of open infrastructure BIM. Buildings, 2023, vol. 13, no 7, p. 1593. Disponible en: https://doi.org/10.3390/buildings13071593

[23] buildingSMART International (2024). Industry Foundation Classes (IFC) – An Introduction. buildingSMART Technical. (Introducción general al estándar IFC, incluyendo su estatus ISO 16739-1:2024). Disponible en: https://github.com/buildingSMART/technical.buildingsmart.org/blob/main/Industry-Foundation-Classes-(IFC).md

[24] BORRMANN, André, et al. Industry foundation classes: A standardized data model for the vendor-neutral exchange of digital building models. Building information modeling: Technology foundations and industry practice, 2018, p. 81-126. Disponible en: https://doi.org/10.1007/978-3-319-92862-3_5

[25] CEROVSEK, Tomo. A review and outlook for a 'Building Information Model'(BIM): A multi-standpoint framework for technological development. Advanced engineering informatics, 2011, vol. 25, no 2, p. 224-244. Disponible en: https://doi.org/10.1016/j.aei.2010.06.003

[26] ISO 16739-1:2024. Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries – Part 1: Data schema. International Organization for Standardization, Geneve, 2024. (Norma internacional que especifica el esquema de datos IFC4.3, sucediendo a ISO 16739:2013).

[27] THEILER, Michael; SMARSLY, Kay. IFC Monitor–An IFC schema extension for modeling structural health monitoring systems. Advanced Engineering Informatics, 2018, vol. 37, p. 54-65. Disponible en: https://doi.org/10.1016/j.aei.2018.04.011

[28] EASTMAN, Charles M. BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors. John Wiley & Sons, 2011. Disponible en: https://books.google.es/books?hl=es&lr=&id=-GjrBgAAQBAJ&oi=fnd&pg=PP7&dq=Eastman,+C.,+Teicholz,+P.,+Sacks,+R.,+Liston,+K.+(2018).+BIM+Handbook:+A+Guide+to+Building+Information+Modeling+(3%C2%AA+ed.).+John+Wiley+%26+Sons.&ots=PhmFf29ojr&sig=Mf2TTSw6NgJBY1mgHpWgytjPcOs

11. Anexos

En este apartado se incluyen los anexos correspondientes al presente Trabajo Fin de Grado para el Grado en Ingeniería Informática de la Universidad San Jorge. En primer lugar, se presenta la propuesta del proyecto final, elaborada conforme a los requisitos previamente establecidos y planificada conjuntamente por el alumno y los directores Manuel Ballarín y Alejandro Giménez. En segundo lugar, se incorporan las actas de las reuniones realizadas durante las fases de diseño, desarrollo y gestión del trabajo, incluyendo aquellas mantenidas con el equipo interno en la empresa. Finalmente, se adjuntará el índice de ilustraciones y tablas y un glosario de términos más relevantes citados en

11.1. Propuesta de proyecto final

Acorde al *FI 030* enviado el miércoles 23 de octubre de 2024 y aprobado el 8 de noviembre del mismo año.

11.1.1. Título del proyecto

Análisis y Desarrollo de un plugin en Unity para la carga y modificación de modelos de datos de construcción en tiempo real.

11.1.2. Descripción y justificación del tema a tratar

Diseño y Desarrollo de un plugin para Unity que facilite la integración y estandarización de modelos BIM (Building Information Modeling) en entornos de realidad virtual (VR), enfocado en mejorar la interoperabilidad entre diferentes plataformas y herramientas de diseño arquitectónico.

El plugin permitirá a los usuarios importar, visualizar y manipular modelos BIM directamente dentro de Unity, respetando las normas internacionales de modelado de construcción, como IFC (Industry Foundation Classes). Además, ofrecerá herramientas para la simplificación de geometrías, la optimización de datos y la revisión de estándares de construcción. Este sistema asegurará que los modelos BIM se ajusten a las normas de construcción locales e internacionales, contribuyendo a la eficiencia en la fase de diseño y construcción.

11.1.3. Objetivos del proyecto

1. *Integración de Modelos BIM en Unity*: Desarrollar una herramienta que facilite la importación de modelos BIM en el entorno de Unity, respetando su geometría, metadatos y estructura, permitiendo su visualización y manipulación en tiempo real.
2. *Interoperabilidad entre Plataformas*: Crear un sistema que garantice la interoperabilidad de los modelos BIM entre diferentes plataformas y herramientas de diseño arquitectónico,

permitiendo una transición fluida y coherente entre el uso de diferentes tecnologías y software.

3. *Facilitar la Colaboración en Proyectos AEC*: Proveer herramientas de colaboración que permitan a múltiples usuarios visualizar y manipular modelos BIM en Unity, favoreciendo un entorno de trabajo colaborativo para los equipos de arquitectura, ingeniería y construcción.
4. *Desarrollo de Digital Twins*: Implementar capacidades que permitan la creación y gestión de Digital Twins, integrando modelos BIM con datos en tiempo real para mejorar la toma de decisiones y la gestión de proyectos durante el ciclo de vida de las infraestructuras

11.1.4. Docente que respalda la propuesta

Manuel Ballarín Naya.

Alejandro Giménez Garulo (Hiberus Advanced Solutions S.L.).

11.2. Actas de reuniones

Durante la fase de diseño y desarrollo del software asociado al presente proyecto, se recuerda cómo se indica en el apartado de Metodología que se han llevado a cabo reuniones diarias ("dailys") los días laborables, a las 9:00 horas, con una duración aproximada de veinte minutos desde octubre hasta la finalización completa del presente proyecto. Estas reuniones han tenido como objetivo identificar y resolver bloqueos, además de proporcionar orientación y seguimiento en el progreso del proyecto. Dichas actas no se muestran en dicho apartado.

11.2.1. Acta de primera reunión

REUNIÓN: 0

Fecha:	10/10/2024		
Hora comienzo:	10:00	**Hora finalización:**	11:16
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Álex Giménez, Mario González		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Revisión manual TFG y fechas**	
2	**Discusión sobre propuestas de ideas para posibles TFG (IoT, BIM, Digital Twins)**	
3	**Propuesta de ajustes o ampliaciones para mejorar**	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Estudio de posibilidades en BIM	24/10/24	Mario González
002			
003			
004			
005			
006			

11.2.2. Acta de segunda reunión

REUNIÓN: 1

Fecha:	14/10/2024		
Hora comienzo:	16:15	**Hora finalización:**	17:07
Lugar:	Food Truck EARTE		
Elabora acta:	Mario González		

Convocados:	Manuel Ballarín, Álex Giménez, Mario González

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Conocernos**	
2	**Fijar idea de TFG**	
3		
	Próxima reunión: 24/10/2024	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Entrega de la propuesta definitiva de TFG	24/10/2024	Mario González
002			
003			
004			
005			
006			

11.2.3. Acta de tercera reunión

REUNIÓN: 2

Fecha:	24/10/2024		
Hora comienzo:	16:00	**Hora finalización:**	16:46
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Álex Giménez, Mario González		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Inicio, guía y bases del proyecto**	
2	**Finalización y confirmación de la propuesta definitiva**	
3		
4		
	Próxima reunión: 14/11/2024	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Envío de la propuesta final en la PDU	24/10/2024	Mario González
002			
003			
004			
005			

006			

11.2.4. Acta de cuarta reunión

REUNIÓN: 3

Fecha:	14/11/2024		
Hora comienzo:	16:00	**Hora finalización:**	17:27
Lugar:	Sala de reuniones 702 - EARTE		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	**Comprensión del manual de TFG y guías sobre índice**	
2		
3		
4		
	Próxima reunión: 28/11/2024	

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Avances en el índice	28/11/2024	Mario González
002	Explicación detallada del software	28/11/2024	Mario González
003			
004			
005			
006			

11.2.5. Acta de quinta reunión

REUNIÓN: 4

Fecha:	28/11/2024		
Hora comienzo:	16:00	**Hora finalización:**	16:58
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	**Aclarar dudas y comprensión del manual + mejoras en el índice**	
2	**Comprensión final de software y diagramas de flujo**	
3		

	Próxima reunión: 19/12/2024	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Investigación guiada por el equipo	19/12/2024	Mario González
002	Avances en gestión de librerías	19/12/2024	Mario González
003			
004			
005			
006			

11.2.6. Acta de sexta reunión

REUNIÓN: 5

Fecha:	3/12/2024		
Hora comienzo:	11:00	**Hora finalización:**	12:20
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Antúnez, Mario González		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Estudio de librerías óptimas**	
2	**Pruebas de rendimiento en la carga de modelos IFC**	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Avances en librerías/software	28/02/2025	Mario González
002			
003			
004			
005			
006			

11.2.7. Acta de séptima reunión

REUNIÓN: 6

Fecha:	5/12/2024		
Hora comienzo:	11:30	**Hora finalización:**	12:06
Lugar:	Zona GLOBE (Planta 2 del edificio de Hiberus)		
Elabora acta:	Mario González		
Convocados:	Jorge Pascual, Mario González		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Avances en la investigación y desarrollo**	
2	**Justificación de elección de motor y librerías**	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Avances en documentación	28/02/2025	Mario González
002			
003			
004			
005			
006			

11.2.8. Acta de octava reunión

REUNIÓN: 7

Fecha:	19/12/2024		
Hora comienzo:	17:00	**Hora finalización:**	18:03
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Comprensión del desarrollo**	
2	**Requerimientos generales en la memoria**	
	Próxima reunión: 28/01/2025	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Mejora general en la redacción de la memoria	28/01/2025	Mario González

002			
003			
004			
005			
006			

11.2.9. Acta de novena reunión

REUNIÓN: 8

Fecha:	28/01/2025		
Hora comienzo:	17:00	**Hora finalización:**	18:38
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	**Guía de metodología acorde al manual**	
2	**Diagrama descriptivo general del software**	
3	**Licitaciones del equipo BIM**	
	Próxima reunión: 14/02/2025	

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Mejoras en la memoria	14/02/2025	Mario González
002	Lectura y comprensión de distintas memorias del equipo	14/02/2025	Mario González
003			
004			
005			
006			

11.2.10. Acta de décima reunión

REUNIÓN: 9

Fecha:	14/02/2025		
Hora comienzo:	14:35	**Hora finalización:**	15:10
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez, Fernando Otal		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Diagramas de flujo**	
2	**Revisión acorde a contenidos del manual y rúbrica**	
	Próxima reunión: 28/02/2025	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Mejoras en el sistema de metodología	28/02/2025	Mario González
002	Mejoras en la redacción académica del marco teórico	28/02/2025	Mario González
003			
004			
005			
006			

11.2.11. Acta de undécima reunión

REUNIÓN: 10

Fecha:	28/03/2025		
Hora comienzo:	14:30	**Hora finalización:**	15:23
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez, Fernando Otal		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Fijar fechas límites de entregas intermedias**	
2	**Revisión general del documento**	
3	**Estado del software**	
	Próxima reunión: 11/04/2025	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Memoria intermedia finalizada	11/04/2025	Mario González
002	Avances finales en el rendimiento del software	11/04/2025	Mario González
003			

004			
005			
006			

11.2.12. Acta de duodécima reunión

REUNIÓN: 11

Fecha:	06/06/2025		
Hora comienzo:	14:30	**Hora finalización:**	14:58
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez, Fernando Otal		

Orden del día / Acta

No.	**Asunto**	**Acuerdo**
1	**Revisión general de la memoria y software**	
2	**Revisión de figura explicativa**	
	Próxima reunión: 12/06/2025	

Resumen de acuerdos

Número	**Acuerdo**	**Plazo**	**Responsable**
001	Memoria prácticamente finalizada	12/06/2025	Mario González
002	Revisión de comentarios en la memoria	12/06/2025	Mario González
003	Diseño final de figura explicativa	12/06/2025	Mario González
004	Mejora de figuras		
005	Mejora en la organización		
006			

11.2.13. Acta de decimotercera reunión

REUNIÓN: 12

Fecha:	12/06/2025		
Hora comienzo:	9:55	**Hora finalización:**	11:04
Lugar:	Sala contigua a SVIT		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez, Fernando Otal		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Revisión general de resolución de comentarios**	
2	**Resolución de conflictos (figuras y organización del documento)**	
	Próxima reunión: 12/06/2025	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Avances finales en el desarrollo de la memoria	17/06/2025	Mario González
002	Resolución completa de comentarios	17/06/2025	Mario González
003	Diseño final de figura explicativa	17/06/2025	Mario González
004			
005			
006			

11.2.14. Acta de decimocuarta reunión

REUNIÓN: 13

Fecha:	17/06/2025		
Hora comienzo:	15:57	**Hora finalización:**	17:24
Lugar:	Microsoft Teams (online)		
Elabora acta:	Mario González		
Convocados:	Manuel Ballarín, Mario González, María Pérez, Fernando Otal		

Orden del día / Acta

No.	Asunto	Acuerdo
1	**Revisión general de cambios finales en la memoria**	
2	**Figura descriptiva**	
	Próxima reunión: 19/06/2025	

Resumen de acuerdos

Número	Acuerdo	Plazo	Responsable
001	Finalización de la figura descriptiva	19/06/2025	Mario González
002	Revisión final de la memoria	19/06/2025	Mario González
003			
004			
005			
006			

11.3. Índice de Ilustraciones

11.4. Índice de Tablas

11.5. Glosario de Términos

AEC: Entorno integrado de Arquitectura, Ingeniería y Construcción que agrupa procesos y herramientas para el diseño y ejecución de proyectos constructivos.

Ad-hoc: Solución diseñada específicamente para un problema o circunstancia particular, sin comprometerse a estándares generales.

Assembly Definition: Archivo de Unity que agrupa y gestiona conjuntos de scripts y recursos en ensamblados independientes para optimizar la compilación.

As-built: Representación del estado real de una construcción tras su finalización, reflejando modificaciones y desviaciones del diseño original.

Big data: Conjunto masivo de datos estructurados y no estructurados cuyo análisis permite extraer patrones y soportar la toma de decisiones.

BIM (Building Information Modeling): Metodología de modelado inteligente que integra geometría y metadatos para gestionar proyectos constructivos en todas sus fases.

CDE (Common Data Environment): Plataforma centralizada para almacenar, compartir y coordinar información de proyectos AEC, evitando duplicidades y errores de versión.

Colliders: Componentes de Unity que definen la forma de colisión de objetos 3D para detectar interacciones físicas en tiempo real.

FBX: Formato de intercambio de datos 3D que almacena geometría, animaciones y metadatos, ampliamente compatible con motores como Unity.

Frustrum: Volumen piramidal de visión de cámara en gráficos 3D que determina qué objetos son renderizados en cada fotograma.

Demelos digitales (digital twins): Réplicas virtuales de activos o sistemas físicos que sincronizan datos en tiempo real para monitorizar y optimizar su rendimiento.

Gráficos fotorrealistas "out of the box": Representaciones visuales con iluminación y materiales avanzados listas para usar sin configuraciones adicionales.

Grafos: Estructuras de nodos y aristas que modelan relaciones y rutas, empleadas en planificación y simulación de recorridos.

HVAC: Sistemas de climatización y ventilación (Heating, Ventilation and Air Conditioning) cuya simulación en BIM mejora el análisis energético.

IFC (Industry Foundation Classes): Estándar abierto para almacenar y transferir información de modelos BIM garantizando interoperabilidad entre software.

Internet of Things (IoT): Conjunto de dispositivos conectados que recopilan y transmiten datos en tiempo real para monitorizar y controlar sistemas físicos.

Integrated Project Delivery (IPD): Enfoque de colaboración temprana en proyectos AEC que integra a todos los agentes para mejorar la eficiencia.

Licencias LGPL/MIT: Tipos de licencias de software libres y permisivas que permiten uso, modificación y distribución con pocas restricciones.

Lean Construction: Filosofía de gestión que maximiza valor y minimiza desperdicios en procesos de construcción mediante principios Lean.

Loops: Estructuras de control que permiten repetir bloques para procesar iterativamente datos o elementos.

Materiales PBR: Modelos de materiales basados en física (Physically Based Rendering) que simulan correctamente las propiedades de luz y superficie.

Motor de videojuegos: Plataforma de software que integra renderizado, física y herramientas de scripting para crear entornos interactivos.

OBJ: Formato de archivo 3D que almacena geometría de mallas y coordenadas de vértices de forma simple y ampliamente soportada.

Parser: Componente que analiza un archivo de texto o datos secuenciales, interpreta su sintaxis y extrae la información estructurada.

Planificación 4D: Extensión de BIM que incorpora la dimensión temporal al modelo 3D para coordinar tareas y secuencias de construcción. (3D + tiempo).

Rendering: Proceso de generar imágenes fijas o animaciones a partir de datos 3D aplicando cálculos de iluminación y materiales.

Retorno de la inversión (ROI): Métrica que relaciona los beneficios obtenidos con el coste de un proyecto para evaluar su viabilidad económica.

Roadmap: Hoja de ruta que describe fases, hitos y objetivos de desarrollo del proyecto o producto a lo largo del tiempo.

Shaders: Programas ejecutados en la tarjeta gráfica que calculan efectos de iluminación, color y texturas para cada píxel o vértice.

Tessellation: Técnica de subdivisión de mallas en tiempo real para aumentar detalle geométrico cuando la cámara se acerca.

Triggers: Componentes de colisión sin respuesta física que detectan eventos de entrada y salida para activar acciones.

Walkthrough: Recorrido interactivo a cámara lenta por un modelo 3D que permite explorar virtualmente un espacio arquitectónico.

Unity: Motor de videojuegos y entorno de desarrollo que facilita la importación, visualización y manipulación de modelos 3D en tiempo real.