UNIVERSIDAD TÉCNICA DEL NORTE

Facultad de Ingeniería en Ciencias Aplicadas

Carrera de Ingeniería en Sistemas Computacionales

MODELO INTERACTIVO DE VISUALIZACIÓN DE INFORMACIÓN UTILIZANDO LIBRERÍAS DE RENDERIZADO 3D EN APLICACIONES WEB, APLICADO A LA REDUCCIÓN DE DIMENSIONES

Trabajo de grado presentado previo a la obtención del título de Ingeniero en Sistemas Computacionales

Autor:

Alejandro Emmanuel Arias Collaguazo

Director:

Ing. Cosme Ortega. MSc

Ibarra – Ecuador

2021

UNIVERSIDAD TÉCNICA DEL NORTE
BIBLIOTECA UNIVERSITARIA

AUTORIZACIÓN DE USO Y PUBLICACIÓN A FAVOR DE LA UNIVERSIDAD TÉCNICA DEL NORTE

1. IDENTIFICACIÓN DE LA OBRA

En cumplimiento del Art. 144 de la Ley de Educación Superior, hago la entrega del presente trabajo a la Universidad Técnica del Norte para que sea publicado en el Repositorio Digital Institucional, para lo cual pongo a disposición la siguiente información:

DATOS DE CONTACTO			
CÉDULA DE IDENTIDAD:	1003964994		
APELLIDOS Y NOMBRES:	Arias Collaguazo Alejandro Emmanuel		
DIRECCIÓN:	Ibarra, Isla San Cristóbal y Ambato #1-40		
EMAIL:	aeariasc@utn.edu.ec		
TELÉFONO FIJO:	545216	**TELÉFONO MÓVIL:**	0998936671

DATOS DE LA OBRA	
TÍTULO:	Modelo interactivo de visualización de información utilizando librerías de renderizado 3D en aplicaciones web, aplicado a la reducción de dimensiones
AUTOR (ES):	Arias Collaguazo Alejandro Emmanuel
FECHA: DD/MM/AAAA	14/09/2021
SOLO PARA TRABAJOS DE GRADO	
PROGRAMA:	[X] **PREGRADO** [ ] **POSGRADO**
TITULO POR EL QUE OPTA:	Modelo interactivo de visualización de información utilizando librerías de renderizado 3D en aplicaciones web, aplicado a la reducción de dimensiones
ASESOR /DIRECTOR:	Msc. Ortega Bustamante Cosme MacArthur

2. CONSTANCIAS

El autor (es) manifiesta (n) que la obra objeto de la presente autorización es original y se la desarrolló, sin violar derechos de autor de terceros, por lo tanto, la obra es original y que es (son) el (los) titular (es) de los derechos patrimoniales, por lo que asume (n) la responsabilidad sobre el contenido de la misma y saldrá (n) en defensa de la Universidad en caso de reclamación por parte de terceros.

Ibarra, a los 14 días del mes de septiembre de 2021

EL AUTOR: **Alejandro Emmanuel Arias Collaguazo**

(Firma).....
Nombre: Alejandro Emmanuel Arias Collaguazo

CERTIFICADO DEL DIRECTOR DE TRABAJO DE GRADO

UNIVERSIDAD TÉCNICA DEL NORTE

FACULTAD DE INGENIERÍA EN CIENCIAS APLICADAS

CERTIFICACIÓN DEL DIRECTOR

Por medio del presente yo Ing. Cosme Ortega, MSc, certifico que el Sr. Alejandro Emmanuel Arias Collaguazo, portador de la cédula de ciudadanía Nro. 1003964994. Ha trabajado en el desarrollo del proyecto de tesis "**Modelo interactivo de visualización de información utilizando librerías de renderizado 3D en aplicaciones web, aplicado a la reducción de dimensiones.**", previo a la obtención del título de Ingeniería en Sistemas Computacionales, lo cual ha realizado en su total responsabilidad.

Es todo cuanto puedo certificar en honor a la verdad.

Atentamente:

COSME MACARTHUR ORTEGA BUSTAMANTE

Firmado digitalmente por COSME MACARTHUR ORTEGA BUSTAMANTE
Fecha: 2021.09.21 08:31:10 -05'00'

Ing. Cosme Ortega, MSc

DIRECTOR DE TESIS

Dedicatorias

«Si eres feliz en este momento no tienes porqué arrepentirte de tu pasado».

Alejandro Arias

Antes de empezar…

Sí, hay una razón para usar esta frase, en realidad dos; aquí expondré una:

Y, esta empieza cuando tomé mi tema de tesis. Un tema que me lo ofrecieron mientras viajaba en un autobús interprovincial a presentarme para las prácticas, esa vez seguro pensé algo como: «¿Por qué no?», y así empezó mi largo camino de trabajo de grado.

Si lo pienso hoy en día, tal vez no fue la mejor forma de escoger mi tema de tesis, y por supuesto fue bastante más difícil de lo que pensé, pero eso no significa que me arrepienta de haber tomado esa decisión bajo la luz intensa del sol atravesando la ventana. Ya qué hoy me siento satisfecho con el resultado mostrado.

Dicho esto (y lamentó empezar tan tarde esta parte, en serio), pienso que si hoy estoy escribiendo esto es por la confianza que muchas personas han depositado en mí.

Así que mi dedicatoria va a todas las personas que confiaron en mí.

Una especial mención a mis padres: Pablo Germán Arias Lara y María Aída Collaguazo quienes confiaron en mí y esperaron que terminara este trabajo. Ya se acabó la espera, sé que ha pasado bastante tiempo y por eso se los dedico a ustedes.

Por último, que dedicatoria estaría completa sin la dedicación a los lectores que están dispuestos a leer este trabajo.

Sin exagerar, este trabajo esta dedicado a ustedes, incluso las correcciones que me iban haciendo era para mejorar la claridad de la lectura, es decir, para que todos pudieran leerlo.

¡Bien!, con eso doy por finalizado esta sección.

Agradecimientos

Por supuesto, a todas las personas que me acompañaron en este camino, y también a aquellas que nunca leerán estas palabras. Por favor déjenme expresarles mis agradecimientos de forma textual: «De verdad, muchas gracias».

Empezar por mis padres es esencial, por todos esos años que me han tenido que soportar hasta que al final llegué a este punto, estoy seguro de que fue un camino largo y difícil para ustedes también, así que muchas gracias.

Un agradecimiento especial a mi tutor: Ing. Cosme Ortega. Nunca olvidaré la clase de cómo funcionan los objetos en programación. Es de lo mejor que aprendí en clases (ya lo dije, espero no arrepentirme en el futuro). Hasta ahora sigo usando esos conceptos en mi día a día, incluso una mañana un compañero lo mencionó casualmente, «Esa clase de objetos es…» y esas cosas. Así que creo que no es solo una opinión mía.

Otro agradecimiento a Neider Requene: sin duda esta tesis no podría existir sin su ayuda, y no estoy exagerando. Donde camina uno, caminan dos, por lo que espero verlo en el futuro, ambos como profesionales.

Siguientes: hay un grupo que seguro no lee estas palabras, y si lo hacen quiero una captura de prueba; yo sabré recompensarlos.

Por último, ustedes queridos lectores. Si solo están de paso les deseo un buen viaje; y si, por el contrario, están dispuestos a leer esta tesis, nos embarcaremos en la profunda falla que existe entre la reducción de dimensiones y los motores de renderizado 3D; prometo curvas.

¡Bien!, creo que eso es todo. De verdad quisiera mencionar a tantas personas y momentos, pero pienso que es mejor dejarlo así.

Dejaré de escribir en este momento.

Hablamos.

Tabla de contenido

Índice de Figuras

Índice de Tablas

Resumen

El proyecto planteado tiene la finalidad de proponer un nuevo modelo interactivo de visualización de información con librería de renderizado 3D en una aplicación web.

Se realizó un marco teórico de modelos interactivos de visualización de información y librerías de renderizado 3D para hacer un modelo interactivo aplicado a la reducción de dimensiones. Este se realizó con una búsqueda mixta entre artículos científicos y bases de datos de confianza de desarrolladores.

Partiendo del marco teórico se seleccionó dos librerías de JavaScript de renderizado 3D: Three.js y Babylon.js. De estas se realizó un *benchmark* en el cual se determina cuál es la mejor librería a nivel de software y hardware. Resultando la librería de Three.js como ganadora.

Con la librería de Three.js la aplicación web del modelo interactivo de visualización de información. La aplicación web permite al usuario establecer coeficientes para la mezcla del *kernel* moviendo una figura geométrica, un dodecaedro; luego, estos coeficientes se envían a un software de reducción de dimensiones donde se aplicará los algoritmos seleccionados por el usuario; finalmente, se obtendrán los resultados que se graficarán en la interfaz.

El modelo interactivo se evaluó con la Característica de Usabilidad de la ISO 25010 a través de un cuestionario SUS, del cual se obtuvo como resultado una calificación de 80.81, que representa Grado A y el Adjetivo de Excelente, por lo que se considera que cumple con la característica de usabilidad.

INTRODUCCIÓN

Antecedentes

A pesar de la existencia de herramientas que alcanzan indicadores de eficiencia en términos de rendimiento computacional, exploración y representación de datos de alta dimensión, carecen de propiedades como interactividad y controlabilidad. Por lo tanto, se requiere una intervención experta que proporcione conocimientos previos al sistema para probar técnicas de reducción de dimensiones (RD), así como interpretar los resultados que no siempre se entienden fácilmente (Pena-Unigarro et al., 2016).

El usuario solo puede establecer parámetros iniciales para obtener los resultados; dentro de este paradigma, incluso los expertos, pueden dedicar mucho tiempo a establecer parámetros destinados a resultados válidos (Diego & G., 2018).

Situación actual

Según (Peña-Unigarro et al., 2017) algunos trabajos han logrado interfaces con métodos de RD con diferentes enfoques y formas de generar mezclas entre los diferentes algoritmos de RD; pero, todos estos trabajos no se enfocan en el diseño de la interfaz, tal son los estudios de (Peluffo-Ordóñez et al., 2015a), (Jose A. Salazar-Castro et al., 2017a), (J. A. Salazar-Castro et al., 2015) o el caso de (Alonso et al., 2015). En su mayoría carecen de propiedades como interactividad y controlabilidad, siendo características importantes del campo de la Visualización de Información (InfoVis) (Therón;, 2017).

Prospectiva

Desarrollar un modelo interactivo de visualización de información que ilustre la información con el propósito de ser más útil e inteligible para el usuario no experto en el campo de la RD. Este modelo interactivo permitirá al usuario generar coeficientes para la mezcla del *kernel* a través de una interfaz; luego, se los enviará a un software de reducción de dimensiones donde se aplicará los algoritmos seleccionados por el usuario, finalmente se obtendrán los resultados y se representará en una interfaz gráfica.

Este proyecto de investigación se lo realiza con el fin de apoyar a la comunidad de investigación científica sobre temas de modelos interactivos de visualización de información.

Planteamiento del problema

Desarrollar una nueva propuesta de un modelo interactivo de visualización de información con librerías de renderizado en tres dimensiones (3D) en una aplicación web que tenga un alto índice de interactividad y controlabilidad por el usuario, con el fin de que usuarios no expertos sean capaces de realizar la mezcla de algoritmos de reducción de dimensiones.

Para poder definir el diagrama de Planteamiento de Problema se utilizó el instrumento de investigación de identificación y clasificación de problemas (Matriz Vester), ver Fig. 1.

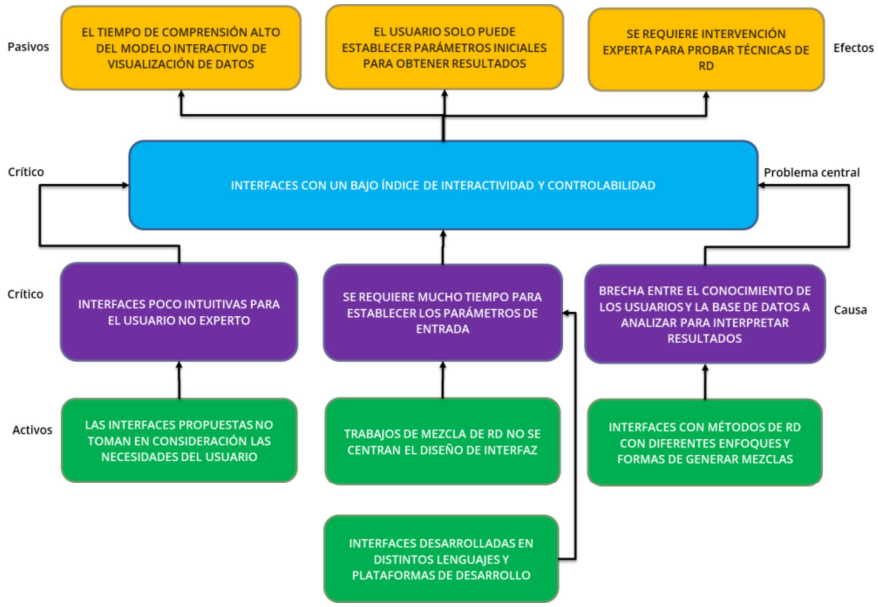


Fig. 1. Diagrama de planteamiento del problema.
Fuente: propio del estudio

Objetivos

Objetivo general

Implementar un modelo interactivo de visualización de información utilizando una librería de renderizado 3D en aplicaciones web aplicado a la reducción de dimensiones.

Objetivos específicos

- Elaborar un marco teórico de librerías de renderizado 3D en entornos web y modelos interactivos de visualización de información propuestos para entornos de reducción de dimensiones.

- Evaluar, definir e implementar librerías de renderizado 3D en un modelo interactivo de visualización de información aplicado a la reducción de dimensiones en entornos web.
- Validar los resultados utilizando la característica de usabilidad de ISO 25010.

Alcance

El proyecto planteado tiene la finalidad de proponer un nuevo modelo interactivo de visualización de información con librería de renderizado 3D en una aplicación web.

Se realizará un marco teórico de modelos interactivos de visualización de información y librerías de renderizado 3D para hacer un modelo interactivo aplicado a la reducción de dimensiones. Que se realizará con una búsqueda mixta entre artículos científicos y bases de datos de confianza de desarrolladores.

Partiendo del marco teórico, se evaluará y definirá las librerías de renderizado 3D que se utilizará en el desarrollo. Se implementará las librerías de renderizado 3D en una aplicación web. Dicha aplicación estará desarrollada en tecnologías web como HTML, CSS, JavaScript bajo la metodología Kanban. La Fig. 2 muestra una aplicación web que permitirá al usuario establecer coeficientes para la mezcla del *kernel*; luego, se los enviará a un software de reducción de dimensiones donde se aplicará los algoritmos seleccionados por el usuario; finalmente, se obtendrán los resultados que se graficarán en la interfaz.

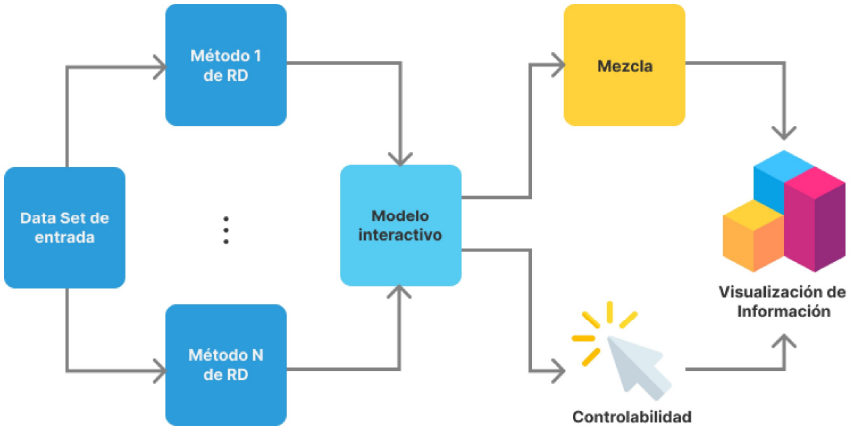


Fig. 2. Diagrama de proceso.
Fuente: propio del estudio

La aplicación web se validará con la característica de usabilidad de la ISO 25010, utilizando las subcaracterísticas:

- Operabilidad.

- Protección contra errores del usuario.
- Estética de la interfaz de usuario.

Justificación

Político

El proyecto se enfoca en el objetivo 9: Industria, innovación e Infraestructura de los Objetivos de Desarrollo Sostenible:

9.4 De aquí a 2030, modernizar la infraestructura y reconvertir las industrias para que sean sostenibles, utilizando los recursos con mayor eficacia y promoviendo la adopción de tecnologías y procesos industriales limpios y ambientalmente racionales, y logrando que todos los países tomen medidas de acuerdo con sus capacidades respectivas.

9.5 Aumentar la investigación científica y mejorar la capacidad tecnológica de los sectores industriales de todos los países, en particular los países en desarrollo, entre otras cosas fomentando la innovación y aumentando considerablemente, de aquí a 2030, el número de personas que trabajan en investigación y desarrollo por millón de habitantes y los gastos de los sectores público y privado en investigación y desarrollo (Naciones Unidas, 2018).

En el Plan Nacional Todo una Vida se enfoca en el Eje 2: Economía al servicio de la sociedad:

Objetivo 5 Impulsar la productividad y competitividad para el crecimiento económico sostenible, de manera redistributiva y solidaria.

Tecnológica

El presente proyecto de titulación tiene como fin contribuir a resolver la problemática presentada en Proyecto Doctoral del Ing. Cosme Ortega, MSc sobre el uso de arquitecturas que optimicen la ejecución de algoritmos de reducción de dimensiones. Implementar esta solución ayudará a la comunidad de investigadores de InfoVis y SDAS GROUP.

Social

El proyecto sirve de apoyo a la comunidad de investigadores que trabajan en áreas de visualización de la información (InfoVis), así como, la comunidad SDAS GROUP (Smart Data Analysis Systems Group), una de sus investigaciones es el

Proyecto Doctoral del Ing. Cosme Ortega, MSc. sobre el uso de arquitecturas que optimicen la ejecución de algoritmos de reducción de dimensiones.

Método de investigación

Se utilizará el método experimental, el que será la guía para realizar el procedimiento de la investigación y así dar respuesta a las interrogantes acerca del desempeño del modelo interactivo de visualización de información, aplicado a la reducción de dimensiones.

Contexto

La Tabla 1 muestra de forma estructurada los diferentes estudios que se realizaron acerca de modelos interactivos de información aplicado a la reducción de dimensiones.

Tabla 1. Repositorios externos

Trabajo	Enlace	Diferencia
El trabajo de (Alonso et al., 2015) llamado "Analysis of Electricity Bill Data using Interactive Dimensionality Reduction".	https://dl.acm.org/doi/10.1145/2797143.2797163	Presentó un caso de estudio donde se propuso una herramienta en HTML5/JavaScript que implementa la reducción de dimensiones interactiva, se utilizó datos de las facturas de luz de la Universidad de León para mostrar a los consumidores herramientas de análisis avanzadas más allá del gráfico de barras.
El trabajo de (Peluffo-Ordóñez et al., 2015b) llamado "Geometrical Homotopy for Data Visualization".	https://dial.uclouvain.be/downloader/downloader.php?pid=boreal:168996&datastream=PDF_01	Propuso un modelo que representa una línea como la unión de dos funciones, mientras que la combinación de varias funciones se representa con los lados de un polígono, siendo tres funciones un triángulo, cuatro un rombo, etc.
El trabajo de (J. A. Salazar-Castro et al., 2015) llamado "Interactive Interface for Efficient Data	https://ieeexplore.ieee.org/abstract/document/7330397	Se basó en el de (Peluffo-Ordóñez et al. 2015) donde la interfaz ahora realiza un análisis de componentes

Visualization via a Geometric Approach".		principales a través de un Kernel PCA, lo que permite obtener una representación bidimensional de los datos.
El trabajo de (Pena-Unigarro et al., 2016) llamado "Interactive Visualization Methodology of High-Dimensional Data with a Color-Based Model for Dimensionality Reduction".	https://ieeexplore.ieee.org/document/7743318	Propone un modelo interactivo basado en un espacio de color RGB, donde cada color primario (red (R), green (G) y blue (B)) representan un método particular de RD; mientras, toda la gama de colores derivados de la combinación se reflejará en la mezcla de métodos de RD.
El trabajo de (Jose A. Salazar-Castro et al., 2017b) llamado "Dimensionality reduction for interactive data visualization via a Geo-Desic approach".	https://ieeexplore.ieee.org/document/7885740	Propone una estrategia geodésica para establecer los coeficientes que se combinarán linealmente los métodos de RD.
El trabajo de (Therón;, 2017) llamado "Interactive Data Visualization Using Dimensionality Reduction and Similarity-Based Representations".	https://link.springer.com/chapter/10.1007/978-3-319-52277-7_41	Propone un modelo interactivo basado en unas barras de ecualizador que permiten al usuario establecer los coeficientes para la mezcla del kernel.
El trabajo de (Peluffo-Ordónez, 2018) llamado "Angle-Based Model for Interactive Dimensionality Reduction and Data Visualization".	https://link.springer.com/chapter/10.1007/978-3-030-01132-1_17	Propone un modelo interactivo que está basado en la geometría de un triángulo, usando principalmente la teoría de ángulos internos y externos de la geometría Euclidiana.
El trabajo de (Diego & G., 2018) llamado "A Novel Color-Based Data Visualization Approach Using a Circular Interaction Model and Dimensionality Reduction".	https://link.springer.com/chapter/10.1007/978-3-319-92537-0_64	Propone un modelo interactivo basado en un círculo segmentado por tres líneas separadas a 120 grados en el que cada sección del circulo representa un método de RD.

Fuente: propio del estudio

CAPÍTULO 1

Revisión Bibliográfica

A continuación, se muestra de una manera estructurada el contenido del presente capítulo el cual conceptualiza los puntos esenciales que involucra la creación de un modelo interactivo de visualización de información aplicado a la reducción de dimensiones, ver Tabla 2. Esquematización del marco teórico..

Tabla 2. Esquematización del marco teórico.

Tema principal	Tema secundario	Sumario
Reducción de dimensiones	Qué es un dato	Es una abstracción de una entidad del mundo real.
	Reducción de dimensiones	Representa datos de alta dimensión a baja dimensión, mientras preserva la mayoría de la estructura relevante.
Modelo interactivo de visualización de información	Técnicas de visualización	Basada en pixeles
		De proyección geométrica
		Basada en íconos
		Jerárquicas
Librerías de renderizado 3D en el navegador	WebGL	Librería de gráficos en 3D que permite a los navegadores modernos renderizar escenas en 3D.
	Three.js	Librería de JavaScript basado en WebGL para crear y mostrar objetos en 3D en un navegador web.
	Babylon.js	Librería de renderizado 3D de JavaScript basado en WebGL que permite crear aplicaciones 3D completas y video juegos 3D para la Web.
Librerías de visualización de información	D3.js	Librería de JavaScript permite vincular datos arbitrarios a un objeto Document Object Model (DOM) y luego aplicar transformaciones basadas en datos.
	Plotly.js	Desarrollado sobre D3.js y Stack.gl: es una librería de gráficos declarativos de alto nivel, incluidos gráficos en 3D.

Fuente: propio del estudio

1.1. Reducción de dimensiones

El punto de partida para entender qué es la reducción de dimensiones es conocer qué es un dato, desde aquí se escalará a través de los datos dimensionales, conjuntos de datos multidimensionales y sus inconvenientes hasta finalmente llegar a la reducción de dimensiones como una forma de resolverlos.

1.1.1. Qué es un dato

Un dato o una pieza de información es una abstracción de una entidad del mundo real; por ejemplo, una persona, objeto o evento. Junto con el término dato, los términos: variable, característica y atributo denotan una abstracción individual del dato y generalmente son usados sin distinción (John D. Kelleher, 2018).

Generalmente cada entidad es descrita por un número de atributos (John D. Kelleher, 2018); por ejemplo, en la Tabla 3 una computadora podría tener los siguientes atributos: modelo, marca, procesador, tarjeta de video, cantidad de *Random Access Memory* (RAM), precio y muchas más (Samet, 2005).

Una entidad con su conjunto de atributos también se considera un dato multidimensional, donde cada atributo representa una dimensión; siguiendo con el ejemplo anterior, el atributo modelo representa una dimensión por sí mismo y marca una segunda dimensión. En definitiva, los datos multidimensionales son una colección de puntos en un espacio de alta dimensión, es decir más de una dimensión. Entre estos los puntos no solo representan registros generales, como en el caso de la Tabla 3, sino también ubicaciones y objetos en el espacio, donde los atributos usualmente tienen la misma unidad (Samet, 2005).

Generalmente son de dimensiones relativamente bajas; sin embargo, hay muchos campos donde los datos tienen una dimensionalidad considerablemente mayor; es decir, son de alta dimensionalidad. Un ejemplo, que no son datos de ubicaciones, son los datos de imágenes (Samet, 2005). A estos se los conoce como datos no espaciales de alta dimensionalidad (DNEAD).

Un conjunto de datos (*data set*) es una colección de entidades y cada una se describe en términos de un conjunto de atributos. Básicamente, como se muestra en la

Tabla 3. Ejemplo de un conjunto de datos., estos se organizan en una matriz de $n * m$ datos, donde n es el número de entidades o filas, y m es el número de los atributos o columnas (John D. Kelleher, 2018).

Tabla 3. Ejemplo de un conjunto de datos.

ID	Modelo	Marca	Procesador	Tarjeta de video	Cantidad de RAM	Precio
1	ROG Zephyrus G15	Asus	AMD Ryzen 7 4800HS	NVIDIA GeForce RTX 2060	16 GB	1,399.99
2	Creator 15 A10SEV-001	MSI	Intel Core i7-10875H	NVIDIA GeForce RTX 2060	32GB	1,998.05
3	ROG Strix G15	Asus	Intel Core i7-10750H	NVIDIA GeForce RTX 2060	16 GB	1,299.00
4	OMEN 15 Gaming Laptop	HP	Intel Core i7-10750H	NVIDIA GeForce RTX 2070 Super Max-Q	32 GB	1,494.99

Fuente: basado en (Samet, 2005)

1.1.2. Reducción de dimensiones

Hay una discusión relacionada con los DNEAD, esta dice que a mayor dimensionalidad los métodos de indexación ya no recuperan eficientemente los objetos buscados, lo que se le atribuye a la conocida como “maldición de la dimensionalidad” (Samet, 2005).

Por supuesto, esta atribución se basa en la premisa de que se conocen bien las características que describen los objetos; sin embargo, es bastante difícil identificarlas, incluso por expertos en el área. Esto es un problema cuando se intenta representar, o visualizar, estos objetos y donde el número e identidad de las características que se deben estar representando son difíciles de entender (Samet, 2005).

La reducción de dimensiones (RD) es uno de los aproximamientos para hacer los datos más simples y fáciles de entender (Rosero-Rosero et al., 2017) y esta tiene como objetivo representar datos alta dimensión en datos de baja dimensión, mientras se preserva la mayoría de la estructura relevante (Sacha et al., 2017).

Uno de los problemas de manejar grandes cantidades de datos es la complejidad que se deriva de estos, a cuantos más datos más dimensiones y más difícil es el entendimiento de estos de parte del usuario, llegando a volverse incluso abstractos (Rosero-Rosero et al., 2017).

1.2. Modelo interactivo de visualización de información

El progreso tecnológico ha permitido la recolección de cantidades muy grandes de datos. Estos datos vienen de diferentes dispositivos de uso diario como teléfonos

inteligentes, una tarjeta de crédito, redes sociales, aplicaciones de *streaming* en vivo como Netflix o YouTube (Keim, 2002), entre otros que se detallan en la Fig. 3.

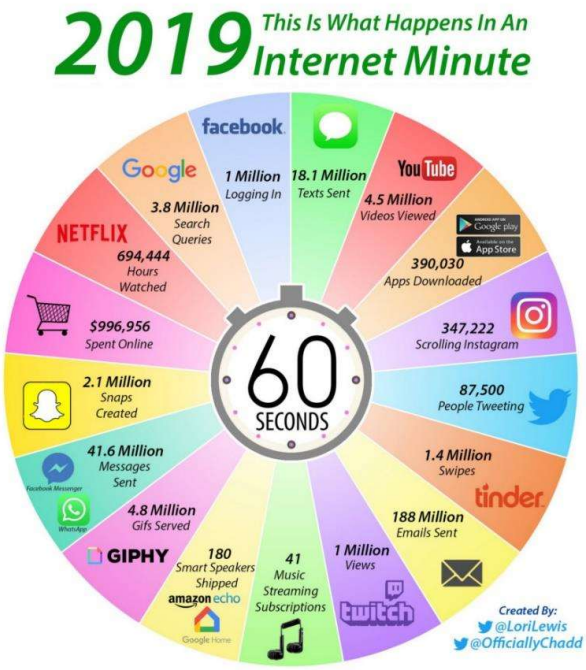


Fig. 3. Datos que se generaban en un minuto en 2019.

Fuente: (Desjardins, 2019)

Usualmente, estos datos son guardados en conjuntos de datos multidimensionales con una alta dimensión y son una fuente potencial de información valiosa; sin embargo, encontrar dicha información es una ardua tarea (Keim, 2002).

Para que esta tarea sea efectiva es importante incluir en la exploración de datos un componente visual. La exploración de datos visual, o visualización de información, tiene como objetivo integrar al ser humano en la exploración de datos, presentando la información de manera visual, permitiendo una mejor percepción de los datos, interactuar con los mismos y hacer sus propias conclusiones (Keim, 2002).

1.2.1. Técnicas de visualización

Las técnicas de visualización de información proveen un alto valor al análisis de la exploración de datos, son especialmente útiles cuando se sabe poco de los datos y las metas de la exploración son vagas (Keim, 2002); estas se encuentran clasificadas de la siguiente forma:

- Técnicas de visualización basada en pixeles.
- Técnicas de visualización de proyección geométrica.

- Técnicas de visualización basada en íconos.
- Técnicas de visualización jerárquicas.

A continuación, se presenta una breve introducción a los conceptos cada una de las técnicas de visualización de información.

1.2.1.1. Técnicas de visualización basada en pixeles

La idea básica es representar el valor una dimensión con el color de un píxel y agruparlos en áreas adyacentes. Estos píxeles permiten la visualización datos en grandes cantidades, incluso por encima del 1'000.000, en una sola ventana (Keim, 2002).

Se utilizan generalmente para mostrar datos ordenados según la consulta requerida (Agarwal, 2014). En la Fig. 4 se muestran varios casos, uno por cada dimensión de los datos, de visualización de información con esta técnica.

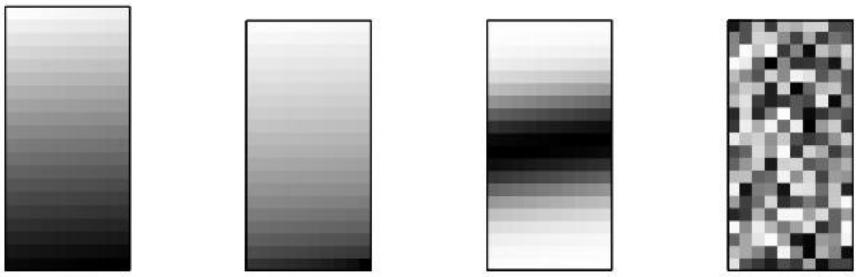


Fig. 4. Ejemplo de visualización de cuatro dimensiones basada en píxeles.
Fuente: (Agarwal, 2014)

Son especialmente útiles para mostrar datos que están ordenados naturalmente de acorde a un atributo (Keim & Kriegel, 1996); sin embargo, llenar la ventana de datos de forma lineal no dará los mejores resultados. Esto es debido a la distancia entre los píxeles (Agarwal, 2014). Una forma de verlo es tomar cada fila y alienarlas en una sola fila horizontal de tal forma que cada fila esté detrás de otra. Con este patrón se puede ver que en el ejemplo de la Fig. 4 la distancia entre el último píxel de una fila y el primero de la siguiente es demasiado grande, a pesar de que ambos se encuentran juntos en la fila que tiene todos los píxeles. Del mismo modo, un píxel que está debajo de otro parecen estar juntos, pero en realidad están muy distantes en el orden global (Agarwal, 2014).

Una forma de resolver este problema es colocar los datos en base a la distancia y no en base al valor. Hay dos acercamientos a este paradigma.

a) En una disposición en espiral generalizada de un atributo, por ejemplo: *Snake-Spiral*, *Peano-Hilbert-Spiral*, *Morton-Spiral*; o
b) en disposición general de ejes de un atributo, por ejemplo: *Snake-Axes*, *Peano-Hilbert-Axes*, *Morton-Axes*, tal como se muestran en la Fig. 5.

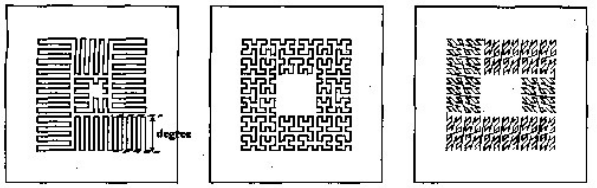


Fig. 2. Generalized spiral arrangement of one attribute (degree of local pattern is 8): (a) Snake-Spiral; (b) Peano-Hilbert-Spiral; (c) Morton-Spiral.

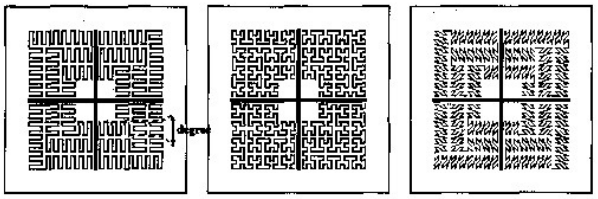


Fig. 3. Generalized axes arrangement of one attribute (degree of local pattern is 4): (a) snake-Axes; (b) Peano-Hilbert-Axes; (c). Morton-Axes.

Fig. 5. Algunas disposiciones generales de ejes y espirales de un atributo.
Fuente: (Keim & Kriegel, 1996)

1.2.1.2. Técnicas de visualización de proyección geométrica

El objetivo de esta técnica es encontrar proyecciones de conjuntos datos multidimensionales que puedan ser visualizados en pantallas en dos dimensiones (2D) (Agarwal, 2014; Keim & Kriegel, 1996). Estas incluyen técnicas de exploración estadísticas como el principal componente de análisis, factores de análisis y escalamiento multidimensional, muchos abarcados bajo el término “búsqueda de proyección” (Keim & Kriegel, 1996).

Los *scatter plot* son utilizados para visualizar estas técnicas en pantalla. Se utiliza un 2D *scatter plot* con un sistema de coordenadas x, y para conjuntos de datos en dos dimensiones; para los conjuntos de datos en tres dimensiones existe el 3D *scatter plot* con un sistema de coordenadas cartesianas de tres ejes o x, y y z. Tanto a los *scatter plot* en 2D como en 3D se pueden extender una dimensión más agregando colores o formas (Agarwal, 2014; Keim & Kriegel, 1996).

Para los conjuntos de datos de cuatro dimensiones o más, los *scatter plots* generalmente son inefectivos. En estos casos el *scatter-plot matrix*, una extensión del *scatter plot*, es bastante útil. Éste muestra una matriz en dos dimensiones de $n\ x\ n$ filas

y columnas de *scatter plots,* como si fuera una galería de *scatter plots* (Agarwal, 2014; Keim & Kriegel, 1996).

Las coordenadas paralelas (*parallel coordinates*) se utilizan cuando el *scatter-plot matrix* se vuelve inefectivo, esto sucede cuando aumenta incluso más la dimensionalidad. En esta técnica cada dimensión es representada por una línea polinomial y están separadas en ejes equidistantes que a su vez están ubicados paralelamente a un solo eje, ver Fig. 6. Sin embargo, debido a que las líneas pueden superponerse, incluso esta técnica se ve limitada por el número de dimensiones que se pueden visualizar, llegando a aproximadamente mil elementos (Agarwal, 2014; Keim & Kriegel, 1996).

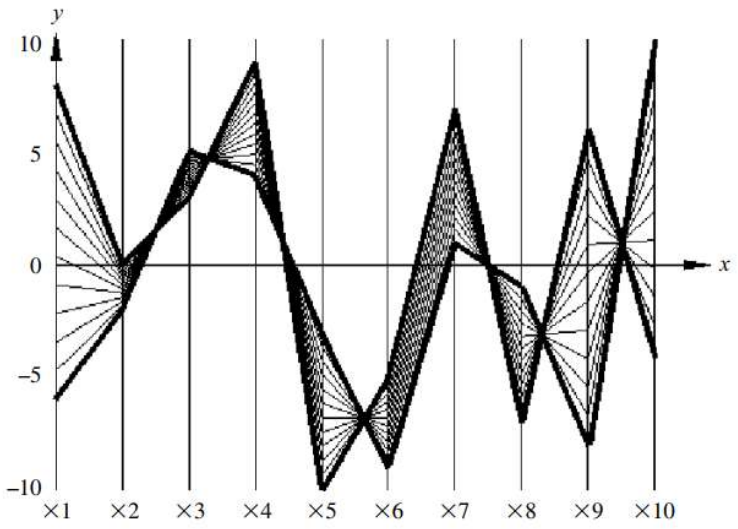


Fig. 6. Ejemplo de visualización de coordenadas paralelas.
Fuente: (Agarwal, 2014)

1.2.1.3. Técnicas de visualización basada en íconos

El concepto base de estas técnicas es convertir una capa multidimensional en un ícono (Keim & Kriegel, 1996). Hay dos acercamientos usuales a estas técnicas, las caras de Chernoff (*Chernoff faces*), Fig. 7.A, y las figuras de palos (*stick figure*), Fig. 7.B (Agarwal, 2014).

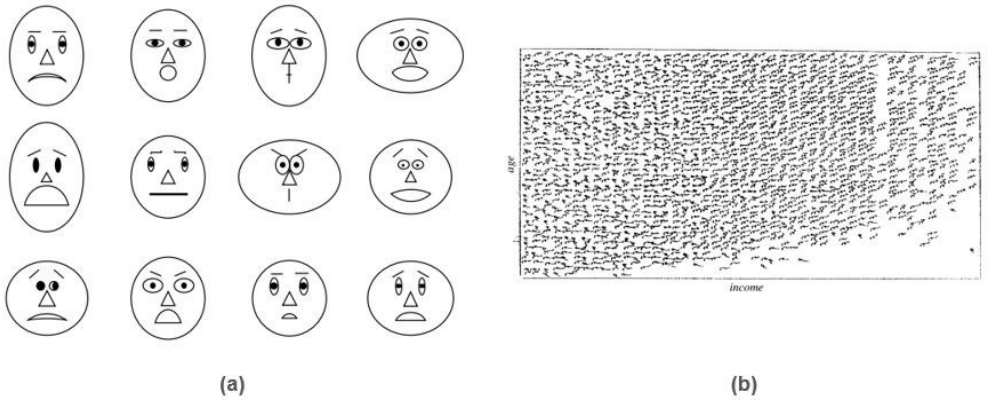


Fig. 7. Ejemplo de visualización basadas en íconos. (a) Ejemplo de las caras de Chernoff. (b) Ejemplo de las figuras de palos.

Fuente: (Agarwal, 2014)

Las caras de Chernoff se aprovecha de la habilidad del ser humano de reconocer las pequeñas diferencias en las características faciales de la cara. Esto lo logra representando los datos multidimensionales en dibujos de caras en espacios de dos dimensiones, partes como la nariz, boca, ojos, orejas son ubicadas por forma y tamaño según el valor de cada dimensión (Agarwal, 2014; Keim & Kriegel, 1996).

Sin embargo, el número de datos que puede mostrar las caras de Chernoff es bastante limitado (Keim & Kriegel, 1996), llegando hasta las 18 variables o dimensiones y utilizando la simetría vertical de la cara se puede llegar hasta las 36 dimensiones (Agarwal, 2014).

Para conjuntos de datos multidimensionales más grandes se puede utilizar las figuras de palos. Al igual que las caras de Chernoff, las figuras de palos se aprovechan, esta vez, de la habilidad del ser humano de reconocer patrones. Como se ve en la Fig. 7B, en esta ocasión, dos dimensiones son utilizadas para ubicar las figuras de palos en el plano y las restantes se representan con los ángulos y/o tamaño de las extremidades de las figuras de palos. Si los datos son lo suficientemente densos, el resultado será un patrón de textura que varía de acorde a las características de los datos (Agarwal, 2014; Keim & Kriegel, 1996).

1.2.1.4. Técnicas de visualización jerárquicas

Las técnicas de visualización jerárquicas se dividen en dos clases. Jerárquicas, que divide las dimensiones en subconjuntos visualizados de forma jerárquica; y las técnicas basadas en grafos, que buscan representar grafos grandes con un gran rendimiento (Keim & Kriegel, 1996).

Técnicas jerárquicas conocidas son: n-Visión, también conocida como “mundo dentro de mundos” (*worlds within worlds*), que subdivide el espacio k-dimensional, se supone de seis dimensiones (6D), en dos subespacios en dos dimensiones, ver Fig. 8; mapas de árbol, que muestran datos jerárquicos como un conjunto de rectángulos anidados; y el apilamiento dimensional (Agarwal, 2014; Keim & Kriegel, 1996).

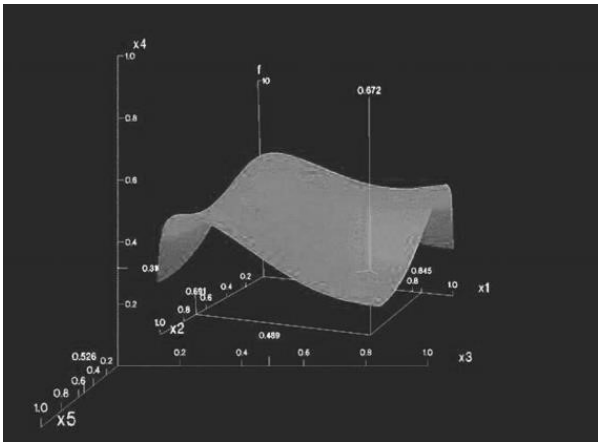


Fig. 8. Ejemplo de la técnica "mundos dentro de mundos".
Fuente: (Agarwal, 2014)

Las técnicas basadas en grafos buscan representar efectivamente grafos grandes usando algoritmos de diseño específicos, lenguajes de consulta, y técnicas de abstracción. Técnicas conocidas son: *Hy+, Margritte, y SeeNet* (Keim & Kriegel, 1996).

1.3. Librerías de renderizado 3D en el navegador

Los modelos de interacción 3D proveen una forma más intuitiva y eficiente de representar la información que el método del plano 2D, ver Fig. 9; y al mismo tiempo, las aplicaciones 3D embebidas en el navegador son más versátiles y portátiles a diferentes dispositivos como computadoras, celulares inteligentes, realidad virtual (VR), entre otras (Li et al., 2020).

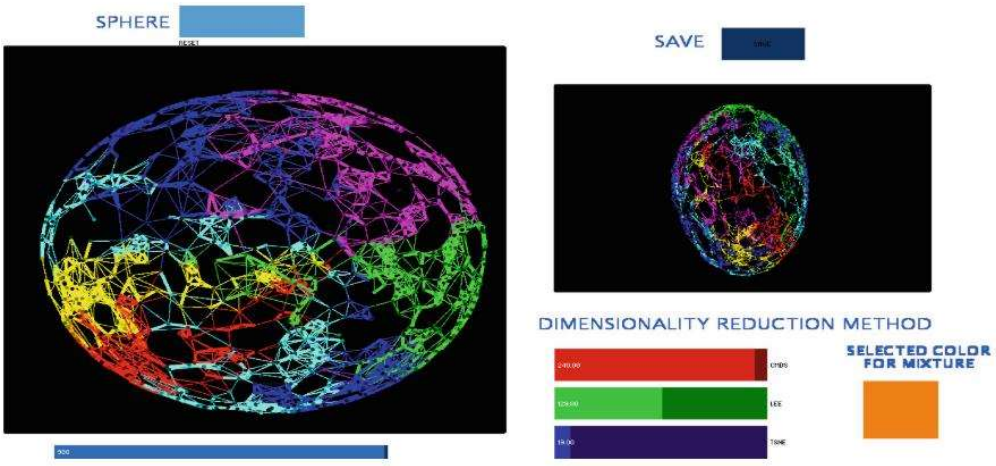


Fig. 9. Ejemplo de modelo interactivo en 3D
Fuente: (Rosero-Rosero et al., 2017)

A continuación, se presenta una introducción a los conceptos que envuelven los modelos de interacción 3D en el navegador. Empezando por *Web Graphics Library* (WebGL) que se ha convertido en el estándar para renderizar gráficos en 3D en el navegador (Gayour & Cantor, 2018).

1.3.1. WebGL

Web Graphics Library, o WebGL, es una librería de gráficos en 3D que permite a los navegadores modernos renderizar escenas en 3D (Gayour & Cantor, 2018), y básicamente se ha convertido en la base de muchos proyectos 3D basados en la web. Dentro de WebGL se encuentra la gestión de gráficos de escenas, animación, sistemas de partículas y muchas más características (Müller et al., 2014).

WebGL utiliza las capacidades de la *Graphic Processing Unit* (GPU) con la cual se hacen cálculos para el renderizado de escenas 3D a través del renderizado basado en *hardware* (*hardware-based rendering*), siendo este mucho más eficiente que su contraparte, el renderizado basado en *software* (*software-based rendering*) ejecutado por la *Central Processing Unit* (CPU) (Gayour & Cantor, 2018).

Para utilizar WebGL en el navegador web se utiliza *Hypertext Markup Language* (HTML) que provee de estructura a la página web, *Cascading Style Sheets* (CSS) el aspecto visual y JavaScript, un lenguaje de programación de propósito general (Angel, 2017; ECMA-International, 2020; World Wide Web Consortium, 2020). Los tres siendo lenguajes propios del navegador web (Müller et al., 2014).

Sin embargo, WebGL por sí mismo solo dibuja puntos, líneas y triángulos; hacer algo útil con WebGL generalmente requeriría bastante tiempo, y código. Aquí es donde Three.js entra en escena, comportándose como una capa de abstracción más que maneja cosas como escenas, luces, sombras, matemáticas en 3D, entre otras (Mirada, 2020).

1.3.2. Three.js

Three.js tiene como objetivo ser una librería de JavaScript liviana y fácil usar con WebGL (Mrdoob, 2020) que se utiliza para crear y mostrar objetos en 3D en un navegador web.

Esta librería permite crear una escenografía con menos líneas de código que con el que se usaría en caso de usar WebGL estándar. Una aplicación hecha con WebGL necesita al menos cuatro componentes distintos, aplicación que se puede realizar con solo uno usando Three.js manteniendo una gran eficiencia (Angel, 2017), ver Fig. 10.

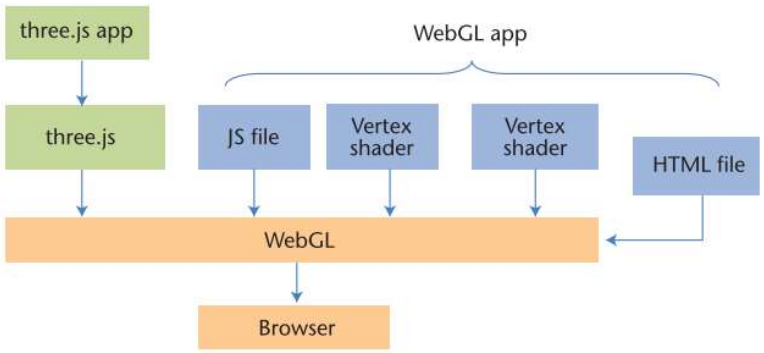


Fig. 10. Diferencias entre una aplicación realizada con WebGL y Three.js
Fuente: (Angel, 2017)

Three.js es capaz de renderizar una escena compleja sin necesidad de obligar al programador acceder a la API de WebGL directamente (Angel, 2017). Esto se debe a la estructura interna y flexible que maneja.

Por ejemplo, una escena básica en Three.js tiene tres objetos básicos. El primero, es el objeto Scene que define la raíz de la escenografía y contiene propiedades como el color de fondo (*background color*) y niebla (*fog*); el segundo, es el objeto Camera que renderiza, o dibuja, la porción de la escena 3D dentro del contenedor (*frustum*) como una imagen 2D en el lienzo (*canvas*); el tercero, el objeto Renderer se encarga de renderizar en el navegador web estos dos objetos (Mirada, 2020), ver Fig. 11.

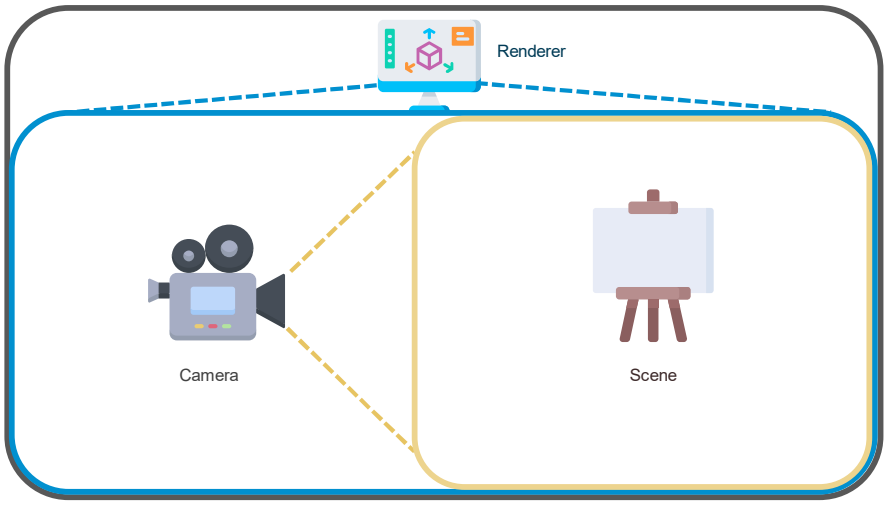


Fig. 11. Estructura de Three.js, los tres objetos básicos.
Fuente: basado en (Mirada, 2020)

Ahora, un objeto Scene puede contener tres objetos: Object3D, provee un conjunto de propiedades y métodos para manipular objetos en un espacio 3D; Light, representa distintos tipos de luces que usan para ver los objetos dentro de la escena; Mesh, representa el dibujo de una geometría con un material específico (Mirada, 2020), ver Fig. 12.

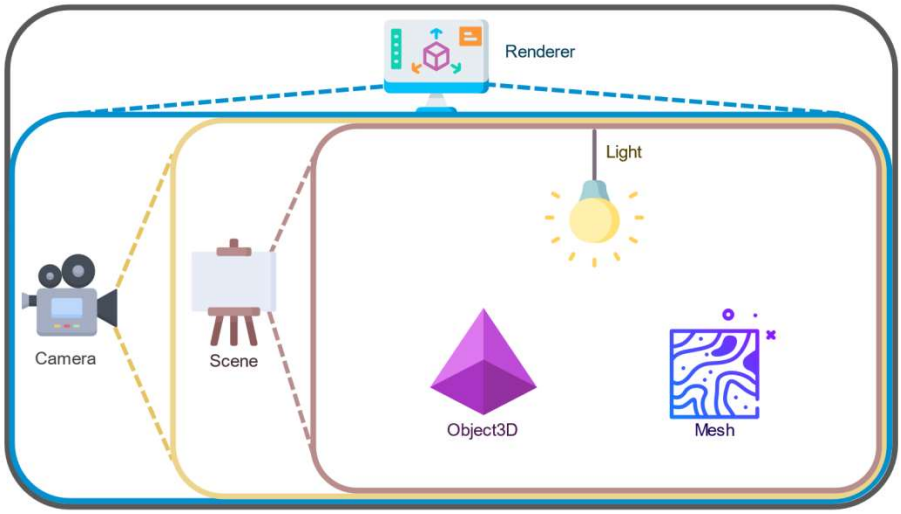


Fig. 12. Estructura de Three.js, desglose del objeto Scene
Fuente: basado en (Mirada, 2020)

Finalmente, cada Mesh es la unión de dos objetos: el objeto Geometry representan los datos de los vértices de alguna figura geométrica como un cubo, esfera, plano, plantas, animales, entre otras; mientras, el objeto Material representan las propiedades de una geometría, tales como el color o el brillo. Un objeto Material puede

hacer referencia a uno o más objetos Texture, que por lo general son imágenes, para envolver con esta la superficie de la geometría (Mirada, 2020), ver Fig. 13.

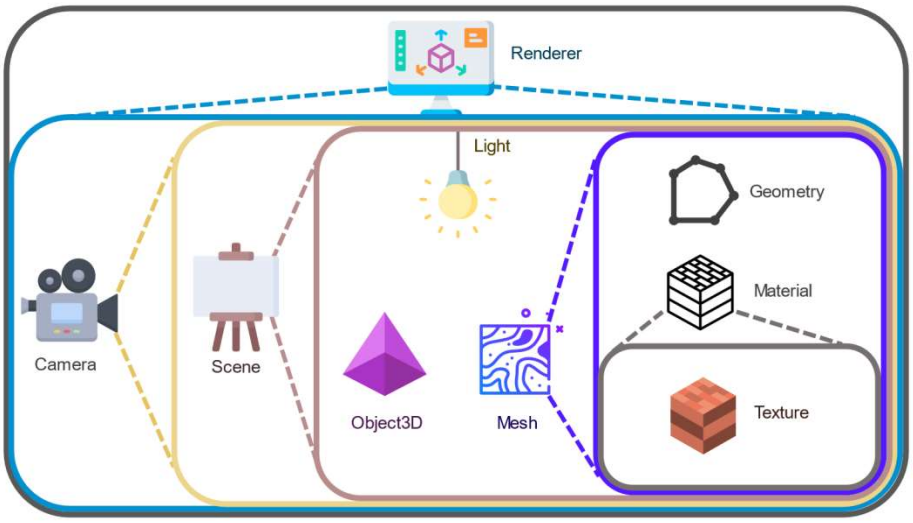


Fig. 13. Estructura de Three.js, desglose del objeto Mesh
Fuente: basado en (Mirada, 2020)

Con esto se ha dado un vistazo general a los conceptos y la estructura básica de una aplicación de Three.js. Una aplicación de Three.js puede ser tan compleja como el programado desee, tal como se puede ver en los ejemplos de (Three.js, 2020a). Sin embargo, no es la única alternativa para crear este tipo de aplicaciones. Por ejemplo, una buena alternativa es Babylon.js.

1.3.3. Babylon.js

Babylon.js es un motor de renderizado 3D de JavaScript basado en WebGL (Catuhe, 2015) que permite crear aplicaciones y videojuegos en 3D para la Web (Moreau-Mathis, 2016).

Babylon.js utiliza los mismos principios de Three.js vistos hasta ahora. El núcleo de Babylon.js es el motor (*engine*), que permite que las escenas (*scene*) puedan crear y gestionar las entidades que se dibujan en la pantalla (Moreau-Mathis, 2016).

Los requerimientos mínimos para crear una escena es una cámara (*camera*) que permite ver las entidades dentro del lienzo y una luz (*light*) que se utilizará para iluminar los objetos en 3D. A diferencia de Three.js, en Babylon.js los objetos 3D se crean a partir de los *Mesh* y no suelen crearse de forma separada. (Moreau-Mathis, 2016).

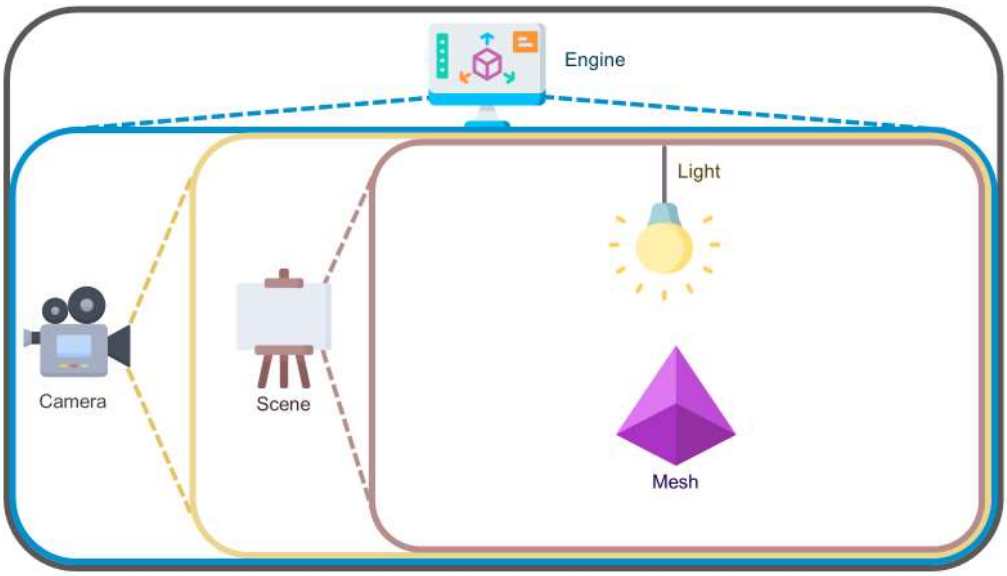


Fig. 14. Escena básica de Babylon.js
Fuente: Basado en (Moreau-Mathis, 2016)

En su forma básica, Babylon.js es sorprendentemente similar a Three.js y se encontrarán pocas diferencias entre ambos en su arquitectura. Es sorprendente la cantidad de aplicaciones que se pueden desarrollar con ambas librerías de renderizado; sin embargo, no son las librerías correctas cuando se trata de visualización de información. Para eso existen mejores alternativas como D3.js.

1.4. Librerías de visualización de información

Debido a su *Application Programming Interface* (API) y también su facilidad y extensibilidad al momento de programar una librería agiliza el proceso de desarrollo de cualquier aplicación.

A continuación, se muestra una lista de librerías de JavaScript orientadas a la visualización de información.

1.4.1. D3.js

D3.js permite vincular datos arbitrarios a un objeto *Document Object Model* (DOM) y luego aplicar transformaciones basadas en datos. Un ejemplo es utilizar D3.js para crear un gráfico de barras interactivo en una etiqueta *Scalable Vector Graphics* (SVG) con transiciones e interacción suaves a partir de una matriz de números (D3, 2020).

D3.js es ideal para visualizaciones en 2D; sin embargo, cuando se trata de visualizaciones de datos en 3D se queda corto por sí solo. Por ejemplo, con D3.js es relativamente sencillo crear una visualización de coordenadas paralelas (Bostock, 2019), ver Fig. 15, pero cuando se trata de una visualización en 3D como un 3D *scatter plot* se vuelve evidente su falta de herramientas. Sin embargo, librerías como D3.js brillan por su extensibilidad y en respuesta a esta carencia la comunidad ha respondido con *plugins* que extienden las funcionalidades de D3.js a nuevas librerías como Plotly.js.

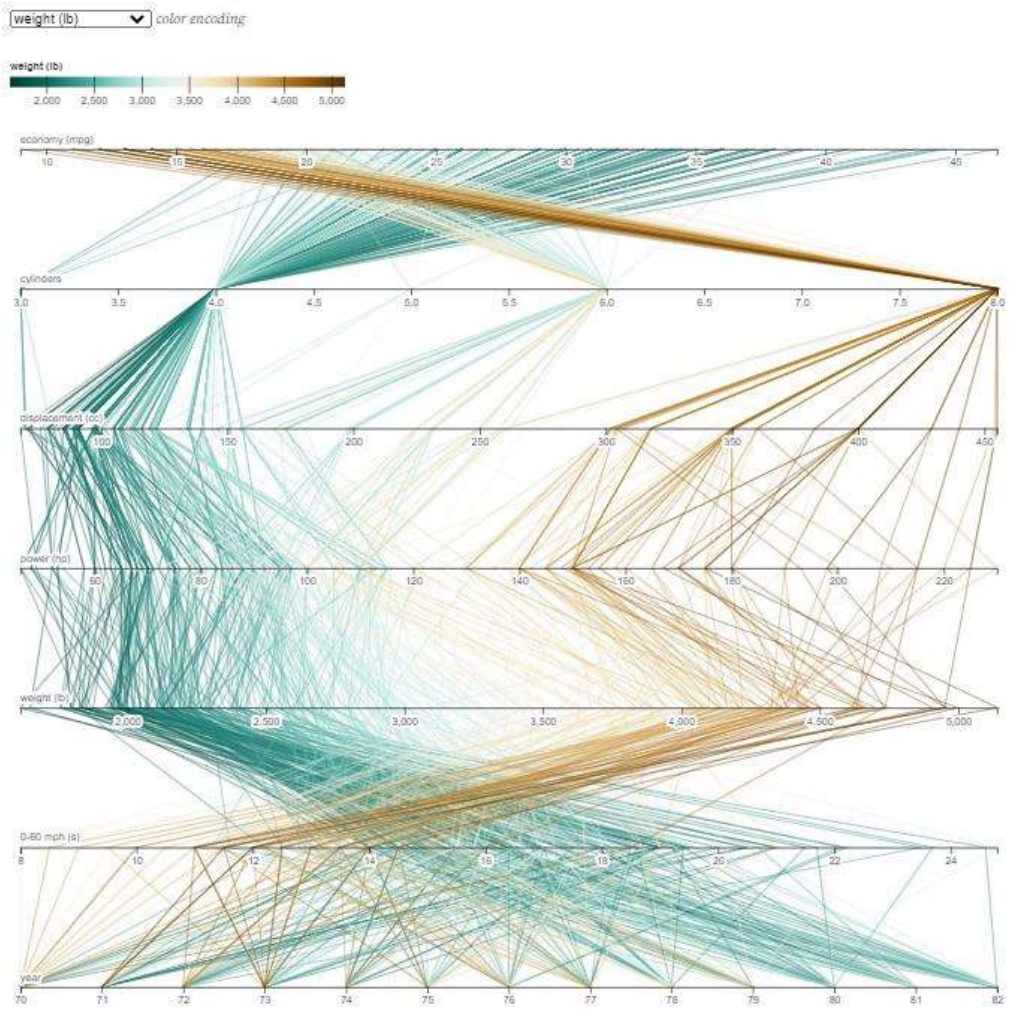


Fig. 15. Ejemplo de la documentación de D3.js de coordenadas paralelas
Fuente: (Bostock, 2019)

1.4.2. Plotly.js

Plotly.js, siendo desarrollado sobre D3.js y Stack.gl es una librería de gráficos declarativos de alto nivel, incluidos gráficos en 3D (plotly, 2020c). Su enfoque está orientado a desarrollar gráficos interactivos para la publicación en línea (plotly, 2020b).

Plotly.js al estar diseñado para renderizar gráficos en 3D se vuelve ideal para la visualización de información resultante de la reducción de dimensiones; por ejemplo, un 3D *scatter plot*, ver Fig. 16.

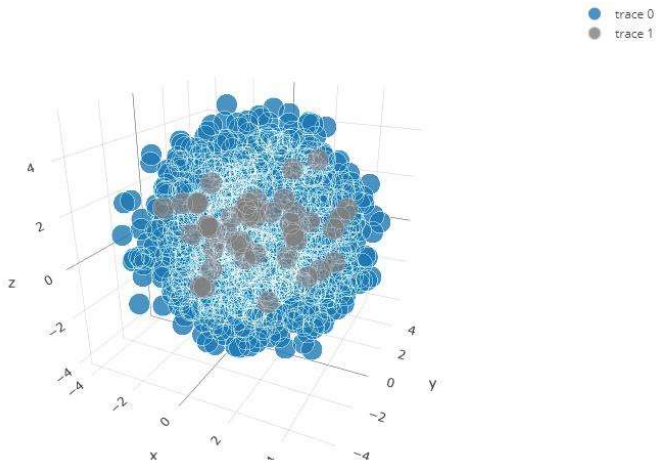


Fig. 16. Ejemplo de la documentación de plotly.js de un 3D scatter plot
Fuente: (plotly, 2020a)

1.5. Librerías de comparativas de rendimiento

A continuación, se muestra una lista de las librerías de JavaScript que se usan en la aplicación encargada de realizar las comparativas de rendimiento de las distintas librerías de renderizado 3D en el navegador.

1.5.1. NodeJS

NodeJS es un *framework* de aplicaciones del lado del servidor, escrito en C++, utiliza el mismo motor de JavaScript que el navegador Chrome llamado V8 (Laksono, 2018; Sterling, 2019).

1.5.2. Webpack

Es una librería, o dependencia, de JavaScript para el *framework* de aplicaciones de NodeJS, se define como un conjunto de módulos estáticos para aplicaciones JavaScript modernas. Internamente genera un gráfico de dependencias, librerías, que mapea cada módulo que el proyecto use activamente y genera uno más paquetes con dichos módulos (Webpack, 2021).

1.5.3. node-os-utils

Es una librería, o dependencia, de JavaScript para el *framework* de aplicaciones de NodeJS, su función es la de un conjunto de utilidades para el sistema operativo. Su

principal utilidad es la de medir el uso de recursos del CPU en punto especifico de tiempo (Wang, 2021).

1.5.4. Systeminformation

Es una librería, o dependencia, de JavaScript para el *framework* de aplicaciones de NodeJS, su función es la de un conjunto de utilidades para recuperar información detallada de hardware y sistema operativo como CPU, RAM, Batería, Gráfica, Discos, etc (Systeminformation, 2021).

1.6. Librerías de análisis de datos

Al igual que las librerías de comparativas de rendimiento que ponen a disposición una API para agilizar el proceso de desarrollo, esta vez se mostrará un listado de librerías de Python orientadas al análisis de datos.

1.6.1. Python

Empezando por Python que es definido como un lenguaje de programación de alto nivel, interpretado y orientado a objetos (Python, 2021). Siendo el cuarto en lista de los lenguajes de programación, *scripting* y lenguajes de marcados más utilizados en el *2020 Developer Survey* de StackOverflow (StackOverflow, 2020).

1.6.2. Numpy

Es una librería de Python utilizada principalmente para las ciencias de la computación, su principal función es el manejo de matrices multidimensionales proporcionando operaciones lógicas y matemáticas como ordenamiento, selección, transformadas de Fourier, algebra lineal, etc (Oliphant, 2010).

1.6.3. Pandas

Es una librería de Python utilizada en las ciencias de la computación construida sobre Numpy, y, a diferencia de este su utilidad más común es el análisis de datos proporcionando herramientas para la lectura y escritura de datos en distintos formatos, indexación o visualización de estos (Pandas, 2020).

1.6.4. Matplotlib

Es una librería de Python, construida sobre Numpy, utilizada para la visualización de datos. Entre el conjunto de herramientas de visualización que ofrece están: gráfico de línea, histogramas, gráfico de dispersión, gráficos en 3D, etc. (Pandas, 2020).

1.7. ISO 25010

La ISO/IEC 25010:2011 es el estándar aprobado por la Organización Internacional de Estandarizaciones, ISO en inglés, para evaluar el modelo de calidad de un producto. Este determina que características se tendrán en cuenta al evaluar un producto de software (ISO/IEC, 2011).

Fig. 17. Características de la ISO/IEC 25010:2011

Este estándar define dos modelos, uno que es para la calidad de producto de software y otro que es para la calidad de uso, cada uno de estos modelos define ciertas características que al ser aplicadas en un producto de software garantizan la calidad de este (ISO/IEC, 2011).

Para este trabajo de las características de la ISO/IEC 2011, solo se evaluarán tres subcaracterísticas: Operabilidad, Protección contra errores del usuario y estética de la interfaz de usuario (ISO/IEC, 2011).

1.7.1. Operabilidad

La subcaracterística de operabilidad es el grado en el que un producto o sistema tiene atributos que lo hacen fácil de operar y controlar (ISO/IEC, 2011).

1.7.2. Protección contra errores del usuario

La subcaracterística de protección contra errores del usuario es el grado en el que un sistema protege a los usuarios de cometer errores (ISO/IEC, 2011).

1.7.3. Estética de la interfaz de usuario

La subcaracterística de estética de la interfaz de usuario es el grado en el que una interfaz de usuario permite una interacción agradable y satisfactoria para el usuario (ISO/IEC, 2011).

CAPÍTULO 2

Desarrollo

La fase de desarrollo del presente proyecto consta de un software desarrollado bajo el marco de trabajo ágil Kanban. El software es una aplicación web que se integrará a través de los servicios de una API con un segundo proyecto desarrollado por otro estudiante de Ingeniería en Sistemas Computacionales.

Kanban es un marco de trabajo de desarrollo de software ágil que busca la comunicación en tiempo real y transparencia total del trabajo. Los elementos de trabajo se representan visualmente en un tablero de Kanban, lo que permite ver el estado actual de las actividades de cada miembro en cualquier momento (Atlassian Agile Coach, 2020).

2.1. Selección de librerías

A continuación, se presentará la metodología utilizada para seleccionar tanto las librerías de renderizado 3D, las cuales se utilizarán para una comparativa de funcionalidad y rendimiento, como la librería de visualización de información, utilizada para graficar los resultados de ambas librerías.

2.1.1. Selección de librerías de renderizado 3D

Para la selección de librerías de renderizado 3D se consideraron dos fases. En la primera fase se aplicaron criterios de inclusión. Los criterios de inclusión considerados fueron: (i) licencia de código abierto, (ii) JavaScript, (iii) TypeScript (superconjunto de JavaScript). Todas las librerías están relacionadas con la categoría de WebGL 2 con contribuciones durante el último año (2019 - 2020) en GitHub.

En la segunda fase se aplicaron criterios relacionados con la relevancia dada por los desarrolladores a cada librería. Las librerías encontradas se ordenaron por el filtro de *Most stars* y se escogieron las dos librerías más relevantes, ver Fig. 18.

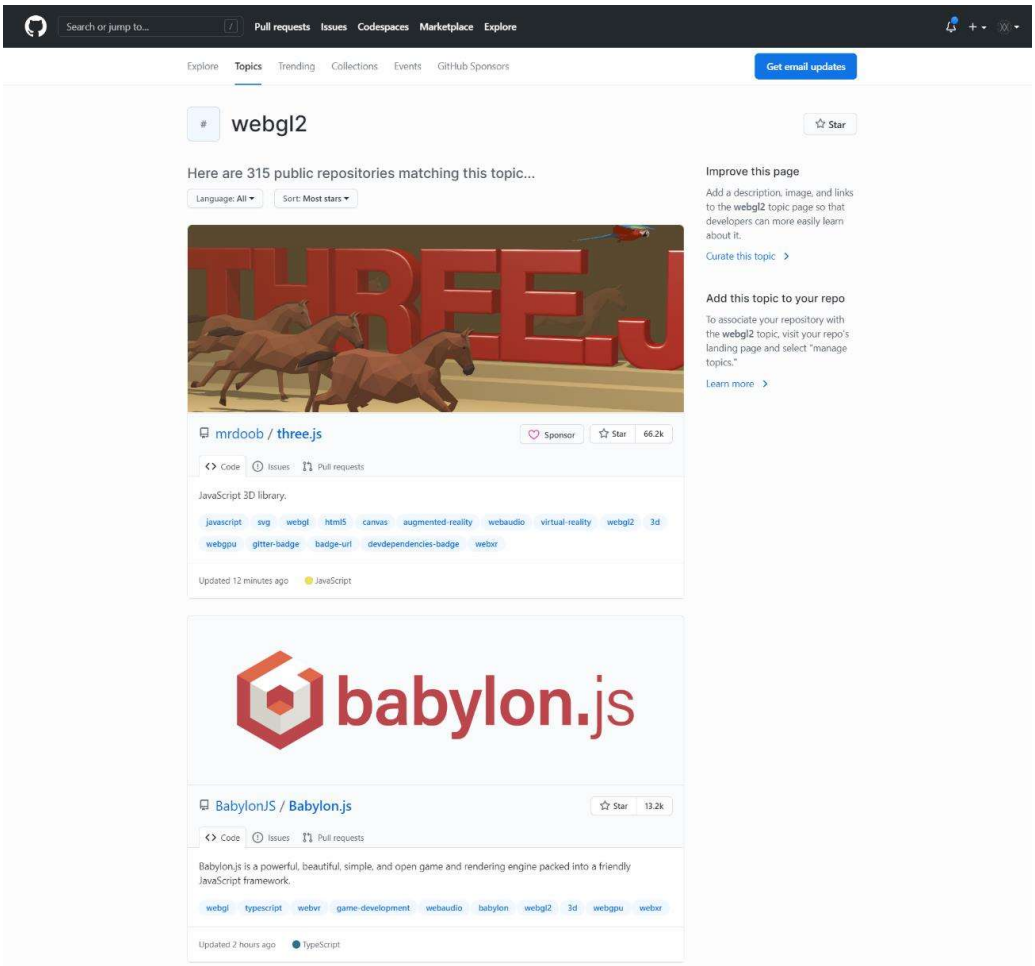


Fig. 18. Librerías de renderizado 3D seleccionadas en la base de datos de GitHub.
Fuente: propio del estudio

Las librerías seleccionadas se escogieron en el mes de diciembre del 2020 en base a datos recopilados hasta esa fecha en GitHub y se presentan en la Tabla 4. Librerías de renderizado 3D seleccionadas..

Tabla 4. Librerías de renderizado 3D seleccionadas.

Nombre de la librería	Lenguaje de programación	Última *release*	*Stars*	Licencia	Soporte WebGL 2
Three.js	JavaScript	24 de diciembre de 2020	66.2 k	MIT	Si
Babylon.js	TypeScript	17 de diciembre de 2020	13.2 k	Apache-2.0	Si

Fuente: basado en datos de (Mrdoob, 2020) y (BabylonJS, 2020a)

2.2. Comparativa de rendimiento de librerías de renderizado 3D

Inmediatamente después de la selección de las librerías de renderizado 3D se procedió a realizar pruebas, aplicaciones simples para probar su funcionalidad, con las librerías seleccionadas.

En estas pruebas se evidenció la simplicidad del modelo elegido, aunque las librerías tienen un campo de aplicación grande, las funcionalidades que se utilizaban en los modelos de visualización no aprovechaban las capacidades de estas, menos una forma clara de compararlas y determinar cuál es la más adecuada para una aplicación de visualización de información.

Motivo por el cual se decidió realizar una prueba de rendimiento específica para comprobar cuál es la más indicada para una aplicación de este tipo.

2.2.1. Métricas de la comparativa de rendimiento

Al seleccionar las métricas del experimento, se buscó que midan el estrés que provoca las librerías sobre el hardware del computador; adicional, se investigó una métrica que mida la eficiencia del navegador renderizando objetos en 3D y al mismo tiempo la experiencia del usuario. Las métricas elegidas para esta comparativa de rendimiento son cuatro:

- Porcentaje de CPU utilizada
- Porcentaje de RAM utilizada
- Porcentaje de GPU utilizada
- *Frames Per Second* (FPS).

CPU, RAM y GPU son las métricas que mejor reflejan el rendimiento a nivel de hardware en una aplicación desarrollada con WebGL, debido a su estrecha relación al momento de ejecutarla.

Por ejemplo, la CPU ejecuta todas las instrucciones del programa, a más complejidad algorítmica mayor es el consumo de esta. A grandes rasgos, en aplicaciones que utilizan WebGL la CPU tiene dos funciones importantes: 1) cargar los datos e instrucciones que ejecutara después en RAM y 2) solicitar a la GPU que renderice en pantalla una escena en 2D o 3D.

Sin embargo, aunque dos aplicaciones muestren resultados similares en estas métricas, la experiencia del usuario puede ser completamente diferente. Esto es más evidente cuando hay una animación, por ejemplo: cuando la animación se detiene por unos milisegundos o varios segundos por completo.

FPS o *Frames Per Second*, hace referencia a la cantidad de fotogramas por segundo que se muestran en pantalla; se utiliza como métrica de rendimiento para animaciones, videos, aplicaciones que utilizan WebGL, etc.

2.2.2. Diseño e implementación de la comparativa

Para realizar la comparativa de rendimiento de ambas librerías es necesario tener un escenario controlado. Iniciando desde el computador en el que se ejecuta las pruebas; además, de una aplicación que pueda ser utilizada con ambas librerías y que sean equivalentes.

2.2.2.1. Dispositivo de prueba

Debido limitaciones de recursos el experimento se lo realizó en un solo computador portátil y, a continuación, se detallan sus especificaciones, ver Tabla 5.

Tabla 5. Características del dispositivo utilizado para el experimento.

Tipo de característica	Característica
Modelo	Asus ROG Zephyrus G15 (2020)
Sistema operativo	Windows 10 Pro
Tarjeta gráfica	NVIDIA GeForce RTX 2060 con diseño Max-Q
Procesador	AMD Ryzen 7 4800HS
Memoria RAM	16 GB

Fuente: propio del estudio.

2.2.2.2. Construcción del aplicativo

Una vez definidas las métricas a analizar y seleccionado el dispositivo en el que se realizará el experimento comenzó la exploración de una aplicación que pueda ser utilizada con ambas librerías; se empezó buscando en los repositorios oficiales para encontrar ejemplos que estresen el hardware del computador.

Por ejemplo, en la documentación de Three.js se puede encontrar aplicaciones de todo tipo, ver Fig. 19; sin embargo, la mayoría de estas son demostraciones de funcionalidades específicas y/o no logran estresar lo suficiente el hardware.

Fig. 19. Aplicación de ejemplo de la documentación de Three.js.
Fuente: (Three.js, 2021)

En contraparte, Babylon.js también tiene su propio conjunto de proyectos en su documentación, ver Fig. 20, e igual que en el caso anterior, son funcionalidades concretas que, en general, no estresan el hardware.

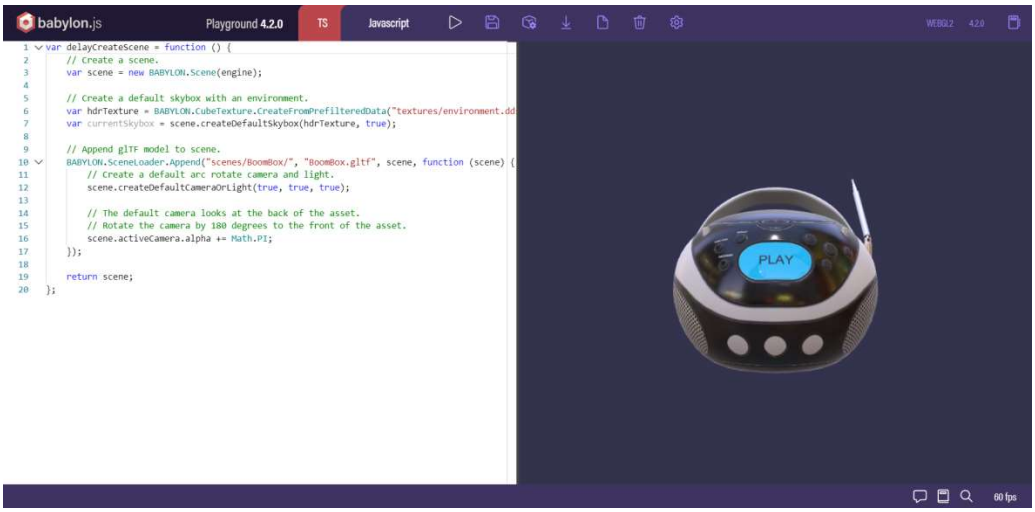


Fig. 20. Aplicación de ejemplo de la documentación de Babylon.js.
Fuente: (BabylonJS, 2020b)

Otro inconveniente con este acercamiento es la diferencia entre las API de Babylon.js y Three.js; lo que hace incompatible el código entre estas, obligando a reescribirlo en su mayoría. De tal forma que, se concluyó que para hacer la comparativa se necesitarían dos aplicaciones que sean equivalentes.

Un ejemplo interesante de proyecto que estresa el hardware del ordenador es una aplicación de Babylon.js, ver Fig. 21, que utiliza un motor de físicas llamado

AmmoJS que se implementa a través de un plugin nativo. Su contrapartida en Three.js es Enable3d, una implementación del mismo motor para esta librería.

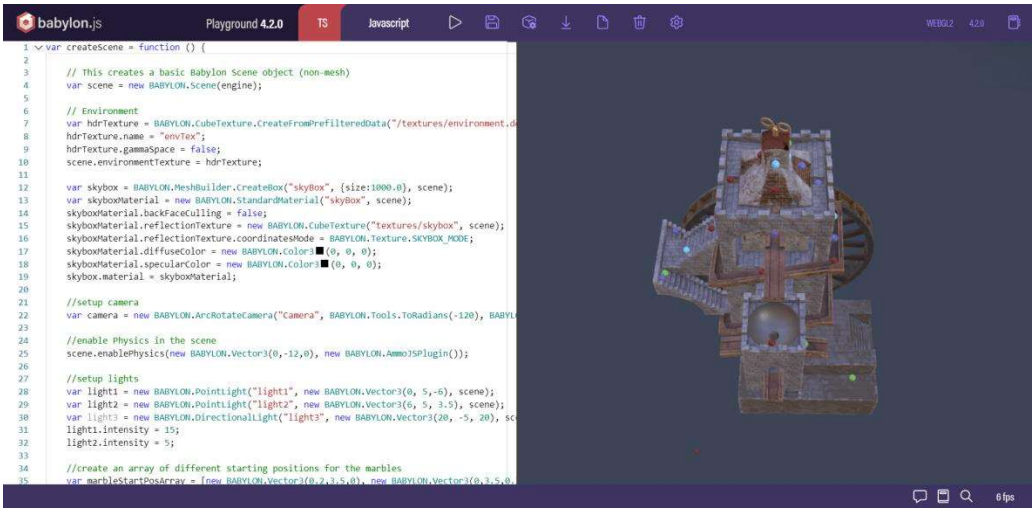


Fig. 21. Ejemplo del motor de físicas AmmoJS en Babylon.js.
Fuente: (BabylonJS, 2020c)

En base a los resultados preliminares se optó por utilizar este acercamiento. Sin embargo, los resultados posteriores fueron desalentadores. Una mala optimización de Enable3d; adicional, el poco control que daba esta sobre la API nativa de Three.js hizo que las aplicaciones no fueran comparables en la práctica.

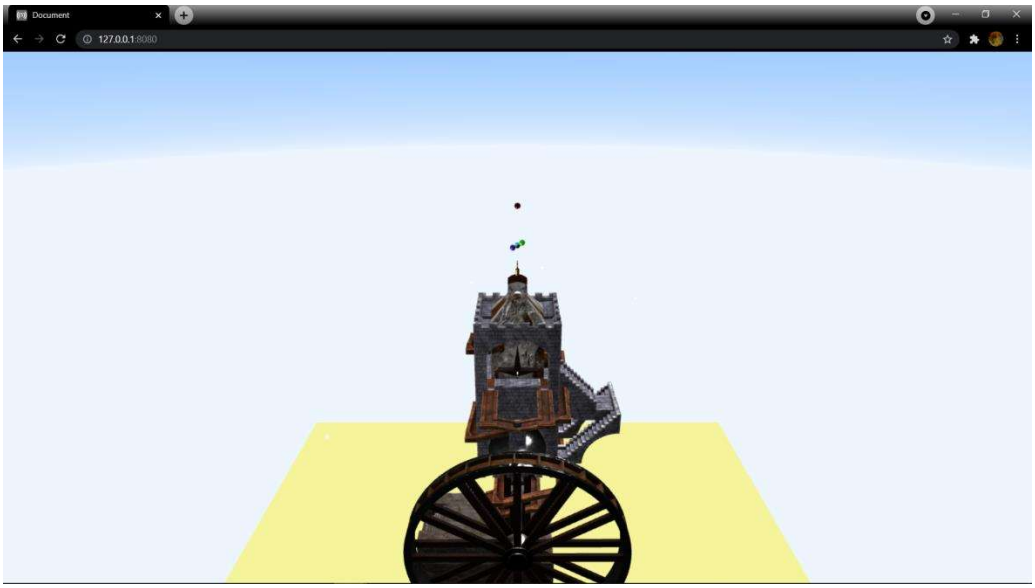


Fig. 22. Adaptación del ejemplo de Babylon.js en enable3d para Three.js.
Fuente: propio del estudio

La aplicación que se usó para la comparativa de rendimiento es una adaptación de un ejemplo de la documentación de Three.js, ver Fig. 23, que busca estresar el hardware renderizando miles de objetos en 3D; además, de animarlos a todos al mismo tiempo.

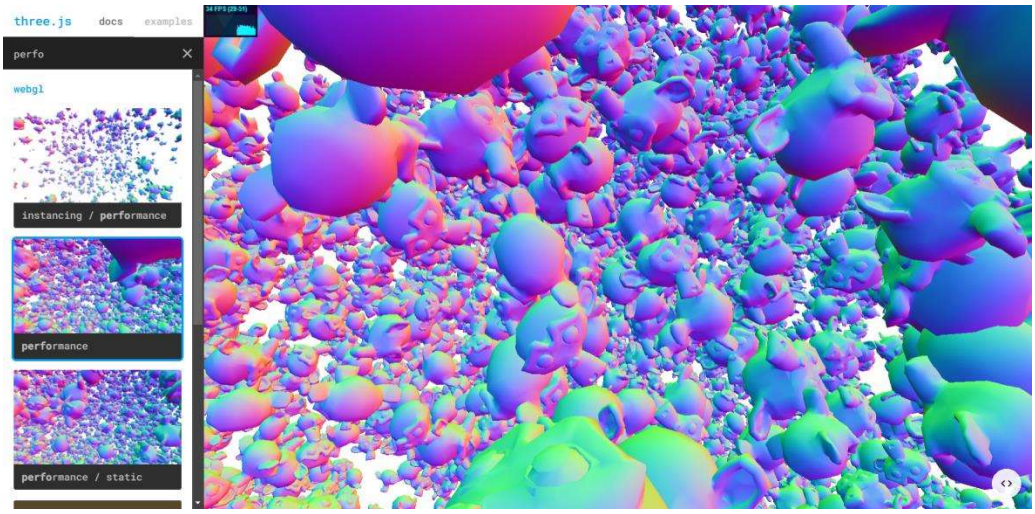


Fig. 23. Ejemplo de Three.js que mide el rendimiento.
Fuente: (Three.js, 2020b)

La versión que se utilizó en la comparativa, ver Fig. 24, descartó los modelos originales, modificó la distancia entre estos y evitó renderizar a todos al mismo tiempo, buscando un acercamiento exponencial en la renderización. Todo esto para que ambas aplicaciones se ejecutaran de forma equivalente.

Fig. 24. Aplicación utilizada en la comparativa, versión Three.js.
Fuente: propio del estudio

La renderización de objetos 3D, también llamados *meshes*, es el principal objetivo de esta aplicación. Con un mínimo de 1 y un máximo de 1097 objetos renderizados de forma aleatoria se buscó estresar el hardware de forma sistemática.

La experiencia de usuario depende de los FPS que se muestren en pantalla, por ejemplo: si está viendo una animación, como unas naves girando en el espacio, su experiencia es peor si la animación no es consistente; tiene retrasos entre *frames* o la imagen se queda congelada por algunos segundos.

Replicar este comportamiento es la razón de no renderizar los 1097 *meshes*, el máximo que puede renderizar Babylon.js antes de congelar la pantalla, desde el comienzo. Cargar progresivamente estos objetos muestran de forma más fiel a un ambiente real la experiencia del usuario; además, de medir el rendimiento de las librerías con distintas cargas de objetos.

Para realizar la carga de *meshes* de forma eficiente se realizó un aproximamiento con una función exponencial, que aumenta la carga en un corto tiempo en gran medida de forma controlada. La función utilizada para estresar el hardware es:

$$f(x) = e^x$$

Donde el e es el número de Euler y x es el número de iteraciones que ejecuta el programa. Se redondeó el resultado para utilizarse en la aplicación.

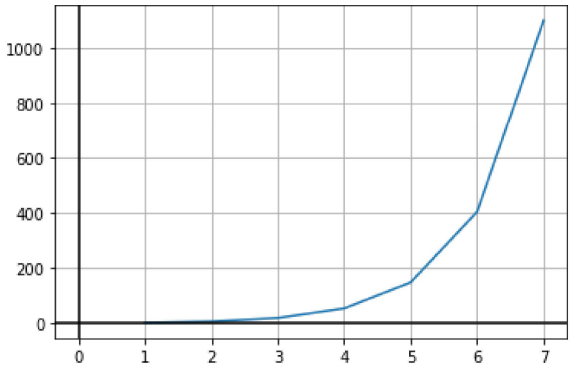


Fig. 25. Gráfica de la función exponencial utilizada en el experimento.
Fuente: propio del estudio

2.2.2.3. Diseño del experimento

Como se mencionó antes, este experimento se lo realizó con la intención de determinar cuál librería de renderizado 3D tiene un mejor rendimiento bajo unas métricas concretas.

El experimento se lo realizó de forma programática, ver Fig. 26, controlado por un *script* diseñado para este fin. Además, de tres aplicaciones que evalúan o registran diferentes partes del experimento.

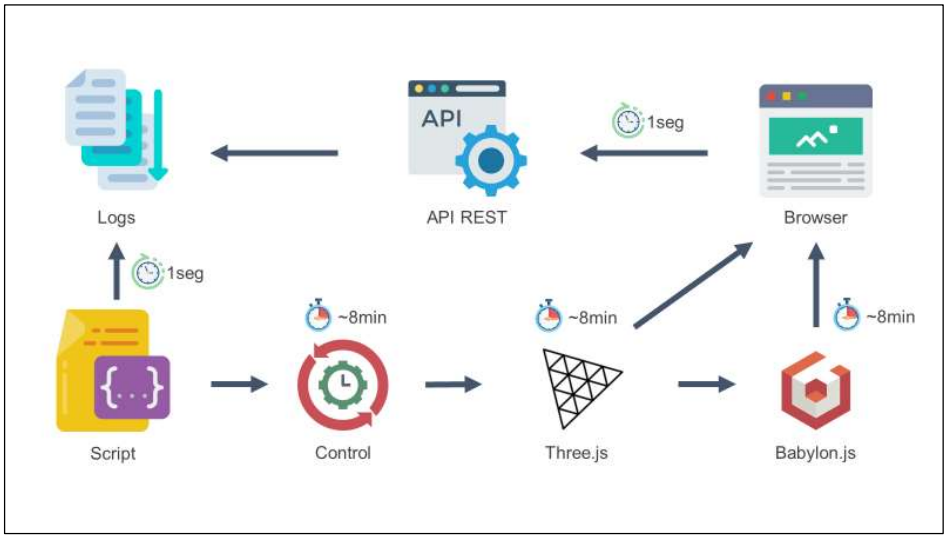


Fig. 26. Diagrama de flujo del experimento.
Fuente: propio del estudio

A modo general, la comparativa de rendimiento se la realiza en tres fases:

1. Inicia un registro de control de las métricas: CPU, RAM y GPU.
2. Inicia el servidor de la aplicación de Three.js y registra las mismas métricas, más FPS.
3. Inicia el servidor de la aplicación de Babylon.js y registra las mismas métricas de antes.

El registro de los resultados se realiza en archivos de texto planos, *logs* en inglés. El Script y la API REST se encargan de este proceso. Hay cinco logs por cada análisis de rendimiento: tres de CPU, RAM y GPU; dos de FPS y *meshes*.

De forma más detallada el flujo del experimento, ver Fig. 26, es el siguiente:

1. Inicia el servidor de la API REST.
2. Inicia la ejecución del script.
3. Después de 1000 milisegundos se inicia un temporizador que se encarga de registrar las métricas CPU, RAM y GPU, fase uno, en un archivo de texto plano cada 1000 milisegundos.
4. 481000 milisegundos después, finaliza el registro de control e inicia el servidor de la aplicación de Three.js, fase dos. Iniciado este se abre la

aplicación en el navegador y comienza el registro de las mismas métricas en un nuevo *log*.

5. El navegador inicia renderizando un *mesh* y cada 60000 milisegundos carga nuevos *meshes* en proporción a la función antes explicada.
6. Al mismo tiempo, cada 1000 milisegundos la aplicación en el navegador envía una solicitud POST a la aplicación API REST con las métricas FPS y *meshes* para que los registre en un nuevo log.
7. 481000 milisegundos después, finaliza el registro de Three.js e inicia el servidor de la aplicación de Babylon.js, fase tres. Abre la aplicación en el navegador y comienza el registro de las métricas en un nuevo *log*.
8. Se repite el paso 5 y 6.
9. 481000 milisegundos después, finaliza el registro de Babylon.js y finaliza la ejecución del *Script*.
10. Finaliza la ejecución del servidor de la API REST.

El navegador que ejecutó cada una de las dos aplicaciones es un punto importante por tratar. Cada navegador es desarrollado de forma diferente, aunque utilicen el mismo motor como Chrome y Edge, esto afecta el rendimiento de la aplicación, ya sea de forma positiva o negativa.

Sin embargo, hacer el experimento en cada uno de los navegadores tomaría un tiempo que excede el establecido para este trabajo. Razón por la cual, se realizó este experimento en tres navegadores: Chrome, el navegador más popular a fecha actual; Firefox, un navegador de la misma lista con un motor diferente al Chromium que utiliza Chrome y similares (Statcounter, 2021); Brave, navegador escogido al azar como control.

2.2.2.4. Diagramas de las aplicaciones

Cada aplicación utilizada en este experimento fue desarrollada de forma separada cumpliendo sus propios requerimientos: sin embargo, las aplicaciones de Three.js y Babylon.js comparten una misma plantilla de proyecto; mientras, REST API y Script son más bien aplicaciones monolíticas.

A continuación, se presentará diagramas de aplicaciones de cada aplicación por separado.

En la Fig. 27 se presenta el diagrama de bloques de la aplicación de Three.js; que se divide entre dos bloques diferenciados: 1) un módulo desarrollado, Webpack Core, con las librerías de Webpack y Babel; y, 2) el módulo de la aplicación de Three.js que se va a utilizar en la comparativa de rendimiento.

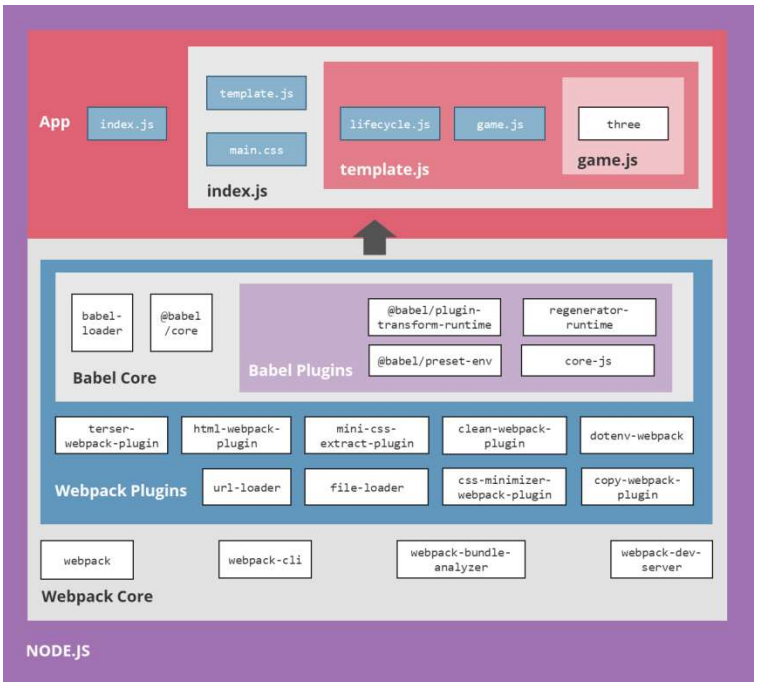


Fig. 27. Diagrama de bloques de la aplicación de Three.js.

Fuente: propio del estudio

En la Fig. 28 se presenta el diagrama de bloques de la aplicación de Babylon.js; que se divide entre dos bloques diferenciados: 1) un módulo desarrollado, Webpack Core, con las librerías de Webpack y Babel; y, 2) el módulo de la aplicación de Babylon.js que se va a utilizar en la comparativa de rendimiento.

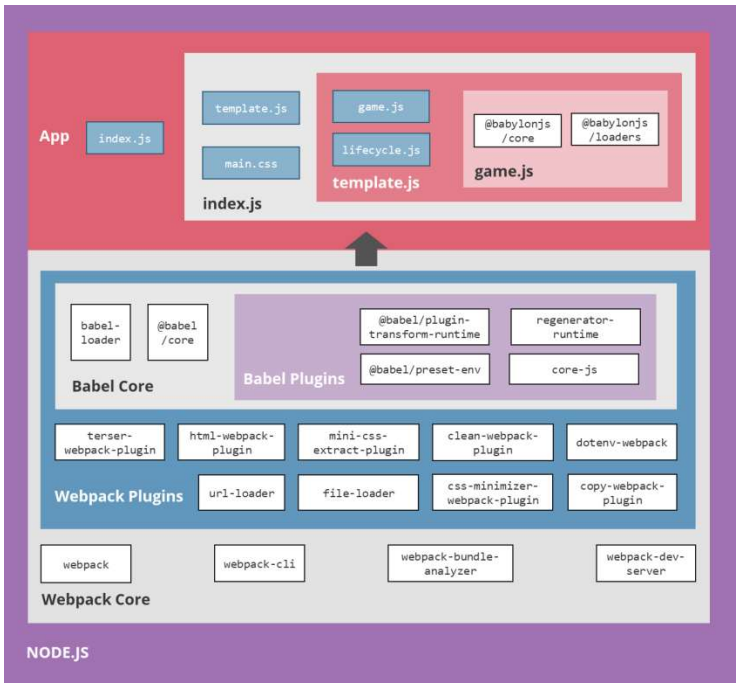


Fig. 28. Diagrama de bloques de la aplicación de Babylon.js.
Fuente: propio del estudio

En la Fig. 29 se presenta el diagrama de bloques de la aplicación de API REST; que se divide entre dos bloques diferenciados: 1) un módulo que va a fungir como API REST, para recibir peticiones GET con los datos de los FPS de la comparativa de rendimiento; y, 2) un módulo de procesamiento para guardar los datos recolectados en archivos de texto plano.

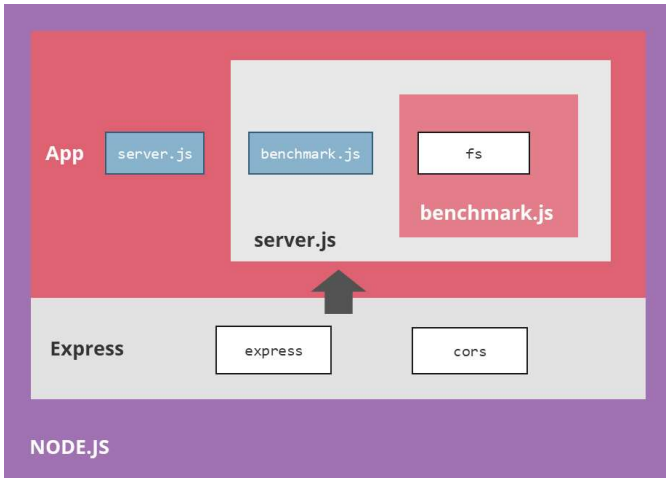


Fig. 29. Diagrama de bloques de la aplicación de API REST.
Fuente: propio del estudio

En la Fig. 30 se presenta el diagrama de bloques del Script utilizado para la automatización del proceso de la comparativa de rendimiento. Dicho script en un programa monolítico, cuya única función, es automatizar la ejecución de la aplicación web desarrolladas en ambas librerías.

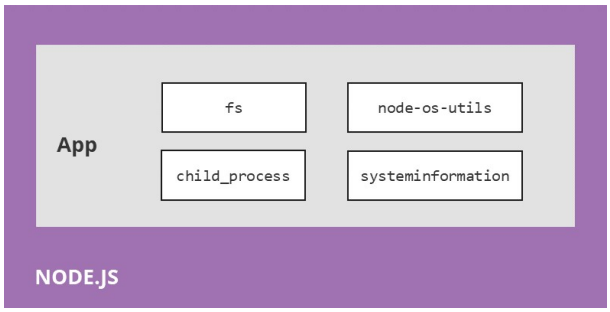


Fig. 30. Diagrama de bloques del Script.
Fuente: propio del estudio

2.2.3. Resultados de la comparativa de rendimiento

Los siguientes datos presentados son el resultado de las pruebas de rendimiento entre las librerías Three.js y Babylon.js. Cada tabla está separada por librería y número de iteración, los datos son una ponderación del promedio de cada iteración. Adicionalmente, todos los datos en crudo recopilados en las pruebas de rendimiento pueden ser encontrados en el Anexo A.

2.2.3.1. Resultados del consumo del *Central Processing Unit* (CPU)

Los datos presentados en las Tabla 6. Promedio de uso de CPU en Chrome., En la Tabla 7 se presenta el promedio del uso de la CPU en el navegador Firefox. Al igual que con Chrome, los datos muestran un menor consumo de recursos de la CPU por parte de Babylon.js en todas las pruebas: con una diferencia porcentual que va desde el 4.4% hasta el 11.6%.

Tabla 7. Promedio de uso de CPU en Firefox. y Tabla 8. Promedio de uso de CPU en Brave. son el promedio de todas las medidas realizadas durante las pruebas de rendimiento en relación con el consumo de CPU. Cabe considerar que, en los siguientes resultados un menor valor de consumo representa un mejor rendimiento.

En la Tabla 6 se presenta el promedio del uso de la CPU en el navegador Chrome. Los datos muestran un menor consumo de recursos de la CPU por parte de Babylon.js en todas las pruebas: con una diferencia porcentual que va desde el 2% hasta el 8.5%.

Tabla 6. Promedio de uso de CPU en Chrome.

Número de iteración	Three.js	Babylon.js
1	97.4%	95.4%
2	81.4%	72.9%
3	76.8%	74.7%

Fuente: propio del estudio.

En la Tabla 7 se presenta el promedio del uso de la CPU en el navegador Firefox. Al igual que con Chrome, los datos muestran un menor consumo de recursos de la CPU por parte de Babylon.js en todas las pruebas: con una diferencia porcentual que va desde el 4.4% hasta el 11.6%.

Tabla 7. Promedio de uso de CPU en Firefox.

Número de iteración	Three.js	Babylon.js
1	64.5%	52.9%
2	65.6%	61.2%
3	67.6%	59.4%

Fuente: propio del estudio.

En la Tabla 8 se presenta el promedio del uso de la CPU en el navegador Brave. Los datos muestran un menor consumo de recursos de la CPU por parte de Babylon.js en dos de tres pruebas, con una diferencia porcentual de: 10.5% en la primera; 10% en la segunda, a favor de Babylon.js; y, con 3.5% a favor de Three.js en la tercera.

Tabla 8. Promedio de uso de CPU en Brave.

Número de iteración	Three.js	Babylon.js
1	83.3%	72.8%
2	87.2%	77.2%
3	69.6%	73.1%

Fuente: propio del estudio.

Los resultados muestran un consumo mayor de recursos de CPU para Three.js en todas las pruebas, ver Fig. 31, Fig. 32 y Fig. 33. Denotando un mejor uso de este aparto por Babylon.js.

La Fig. 31 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Chrome, con relación al porcentaje de uso de la CPU; donde, con un 4.2% Babylon.js muestra un mejor desempeño.

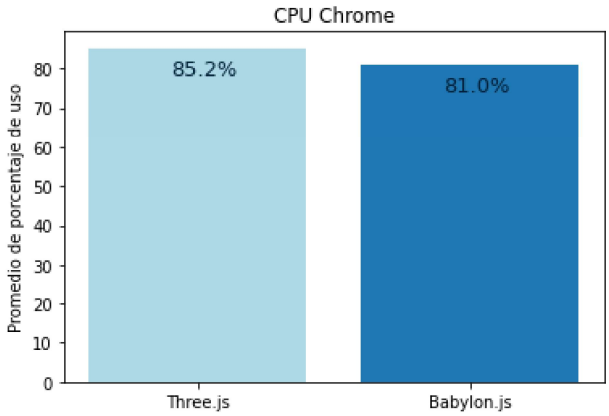


Fig. 31. Promedio de CPU de todos los *benchmarks* pruebas de Chrome.
Fuente: propio del estudio

La Fig. 32 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Firefox, con relación al porcentaje de uso de la CPU; donde, con un 8.1% Babylon.js muestra un mejor desempeño.

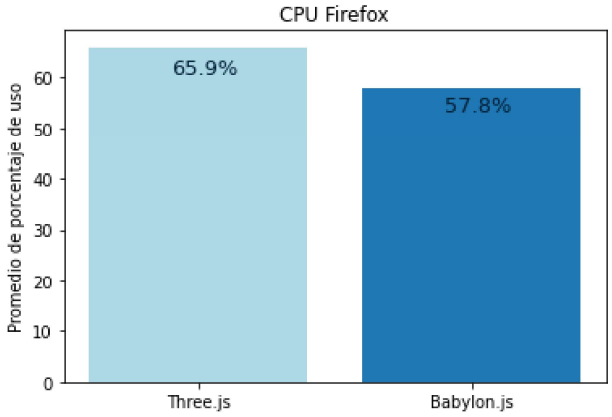


Fig. 32. Promedio de CPU de todos los *benchmarks* pruebas de Firefox.
Fuente: propio del estudio

La Fig. 33 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Firefox, con relación al porcentaje de uso de la CPU; donde, con un 5.6% Babylon.js muestra un mejor desempeño.

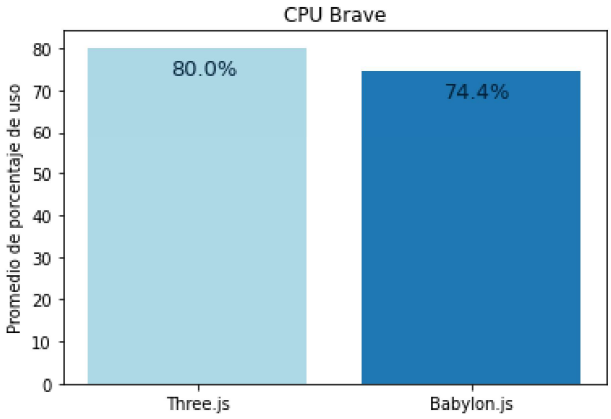


Fig. 33. Promedio de CPU de todos los *benchmarks* pruebas de Brave.

Fuente: propio del estudio

En el promedio general de todos los navegadores: con un 5.9% menor de consumo en CPU, Babylon.js tiene un mejor rendimiento comparado con Three.js, ver Fig. 34.

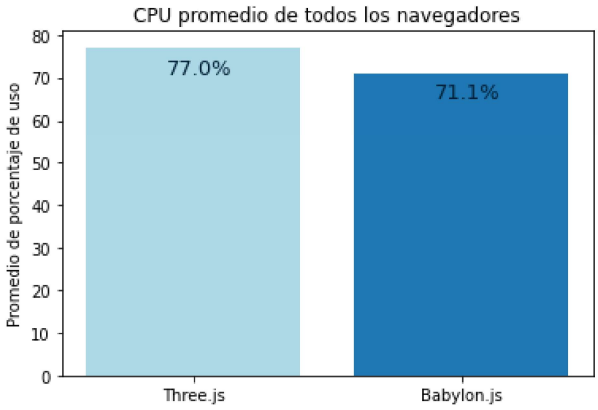


Fig. 34. Promedio de CPU de todos los navegadores.

Fuente: propio del estudio

2.2.3.2. Resultados del consumo del *Graphics Processing Unit* (GPU)

Los datos presentados en las En la Tabla 9 se presenta el promedio del uso de la GPU en el navegador Chrome. Los datos muestran un menor consumo de recursos de la GPU por parte de Three.js en dos de tres pruebas; pero, la diferencia del 0.1% en las dos primeras pruebas no es significativa para determinar la librería que rinde mejor.

Tabla 9. Promedio de uso de GPU en Chrome., Tabla 10. Promedio de uso de GPU en Firefox. y Tabla 11. Promedio de uso de GPU en Brave. son el promedio de todas las medidas realizadas durante las pruebas de rendimiento en relación con el

consumo de GPU. Cabe considerar que, en los siguientes resultados un menor valor de consumo representa un mejor rendimiento.

En la Tabla 9 se presenta el promedio del uso de la GPU en el navegador Chrome. Los datos muestran un menor consumo de recursos de la GPU por parte de Three.js en dos de tres pruebas; pero, la diferencia del 0.1% en las dos primeras pruebas no es significativa para determinar la librería que rinde mejor.

Tabla 9. Promedio de uso de GPU en Chrome.

Número de iteración	Three.js	Babylon.js
1	6.1%	6.2%
2	6.1%	6.2%
3	6.2%	6.2%

Fuente: propio del estudio.

En la Tabla 10 se presenta el promedio del uso de la GPU en el navegador Firefox. Los datos muestran un menor consumo de recursos de la GPU por parte de Three.js en todas las pruebas: con una diferencia porcentual del 0.2%. No es una diferencia significativa para determinar la librería que rinde mejor.

Tabla 10. Promedio de uso de GPU en Firefox.

Número de iteración	Three.js	Babylon.js
1	7.4%	7.6%
2	7.4%	7.6%
3	7.4%	7.6%

Fuente: propio del estudio.

En la Tabla 11 se presenta el promedio del uso de la GPU en el navegador Brave. Los datos no muestran diferencias entre ambas librerías.

Tabla 11. Promedio de uso de GPU en Brave.

Número de iteración	Three.js	Babylon.js

1	6.1%	6.1%
2	6.1%	6.1%
3	6.1%	6.1%

Fuente: propio del estudio.

Los resultados muestran a ambas librerías tener un consumo similar en todas las pruebas, ver Fig. 35, Fig. 36 y Fig. 37.

La Fig. 35 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Chrome, con relación al porcentaje de uso de la GPU; donde, con un 0.1% Three.js muestra un mejor desempeño.

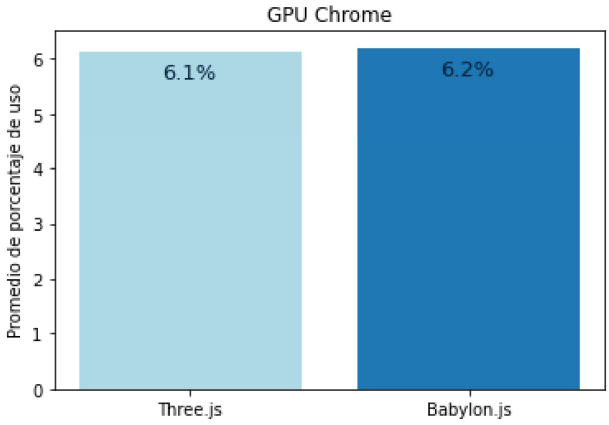


Fig. 35. Promedio de GPU de todos los *benchmarks* pruebas de Chrome.
Fuente: propio del estudio

La Fig. 36 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Firefox, con relación al porcentaje de uso de la GPU; donde, con un 0.2% Three.js muestra un mejor desempeño.

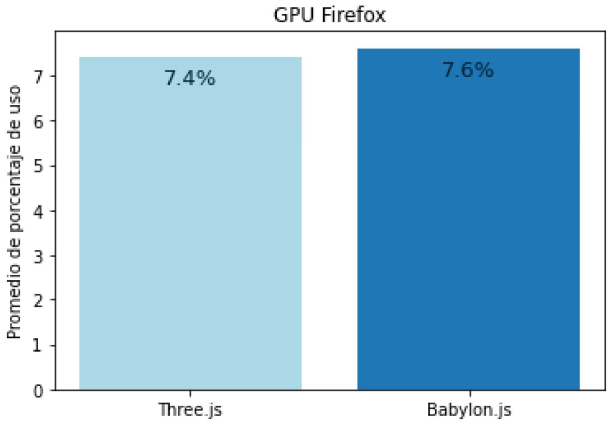


Fig. 36. Promedio de GPU de todos los *benchmarks* pruebas de Firefox.

Fuente: propio del estudio

La Fig. 37 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Brave, con relación al porcentaje de uso de la GPU; donde, no hay diferencias porcentuales entre ambas librerías.

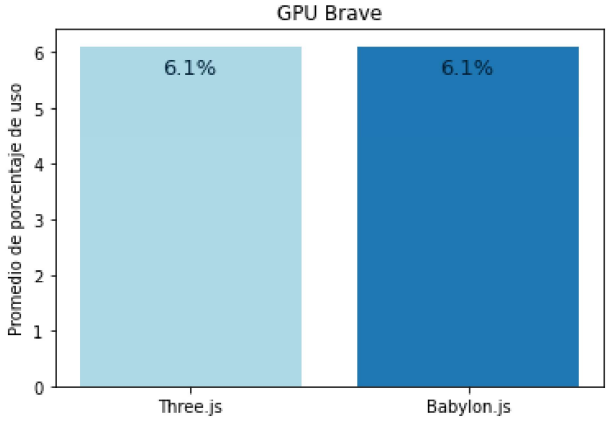


Fig. 37. Promedio de GPU de todos los *benchmarks* pruebas de Brave.

Fuente: propio del estudio

En el promedio general de todos los navegadores: la diferencia de 0.1% en el promedio general del porcentaje de uso de GPU a favor de Three.js no representa una diferencia estadística significativa, ver Fig. 38.

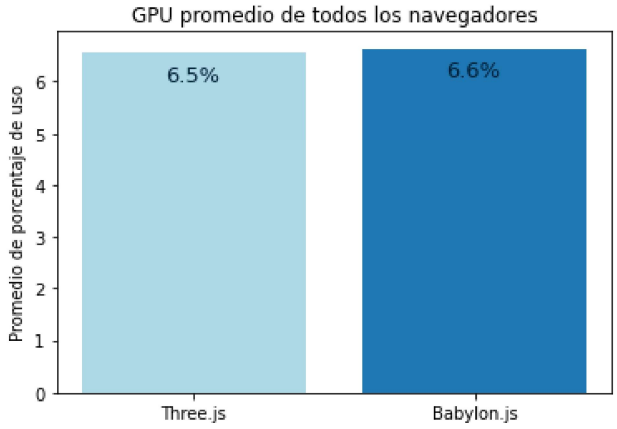


Fig. 38. Promedio de GPU de todos los navegadores.
Fuente: propio del estudio

2.2.3.3. Resultados del consumo de *Random Access Memory* (RAM)

Los datos presentados en las Tabla 12. Promedio de uso de RAM en Chrome., Tabla 13. Promedio de uso de RAM en Firefox. y Tabla 14. Promedio de uso de RAM en Brave. son el promedio de todas las medidas realizadas durante las pruebas de rendimiento en relación con el consumo de RAM. Cabe considerar que, en los siguientes resultados un menor valor de consumo representa un mejor rendimiento.

En la Tabla 12 se presenta el promedio del uso de RAM en el navegador Chrome. Los datos muestran un menor consumo de recursos de RAM por parte de Babylon.js en dos de tres pruebas, con una diferencia porcentual de: 6% en la primera; 0.1% en la tercera, a favor de Babylon.js; y, 1.5% a favor de Three.js en la tercera.

Tabla 12. Promedio de uso de RAM en Chrome.

Número de iteración	Three.js	Babylon.js
1	67.4%	61.4%
2	36.4%	37.9%
3	47.2%	47.1%

Fuente: propio del estudio.

En la Tabla 13 se presenta el promedio del uso de RAM en el navegador Firefox. Los datos muestran un menor consumo de recursos de RAM por parte de Three.js en todas las pruebas: con una diferencia porcentual que va desde el 0.9% hasta el 1.4%.

Tabla 13. Promedio de uso de RAM en Firefox.

Número de iteración	Three.js	Babylon.js
1	36.1%	37.5%
2	36.5%	37.4%
3	36.7%	38.1%

Fuente: propio del estudio.

En la Tabla 14 se presenta el promedio del uso de la RAM en el navegador Brave. Los datos muestran un menor consumo de recursos de la RAM por parte de Three.js en todas las pruebas: con una diferencia porcentual que va desde el 0.2% hasta el 5.5%.

Tabla 14. Promedio de uso de RAM en Brave.

Número de iteración	Three.js	Babylon.js
1	44.4%	46.7%
2	45.8%	46.0%
3	40.5%	46.0%

Fuente: propio del estudio.

Con pequeñas variaciones, ambas librerías tienen un rendimiento similar en el consumo de RAM, ver Fig. 39, Fig. 40 y Fig. 41; por ejemplo: en Chrome Babylon.js muestra un mejor rendimiento; mientras, en Firefox y Brave se decanta a favor de Three.js. Sin embargo, las diferencias son poco significativas para concluir que una es mejor que otra.

La Fig. 39 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Chrome, con relación al porcentaje de uso de la RAM; donde, con un 1.5% Babylon.js muestra un mejor desempeño.

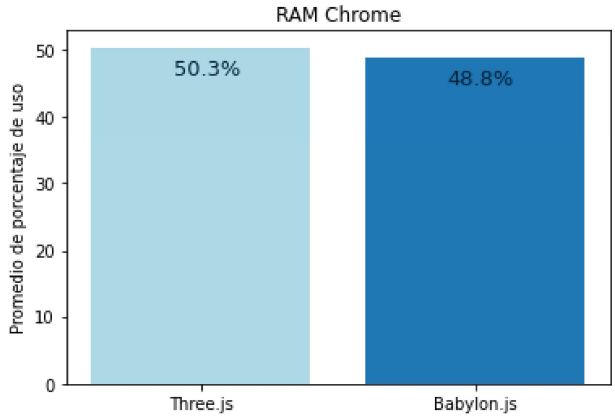


Fig. 39. Promedio de RAM de todos los *benchmarks* pruebas de Chrome.
Fuente: propio del estudio

La Fig. 40 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Firefox, con relación al porcentaje de uso de la RAM; donde, con un 1.3% Three.js muestra un mejor desempeño.

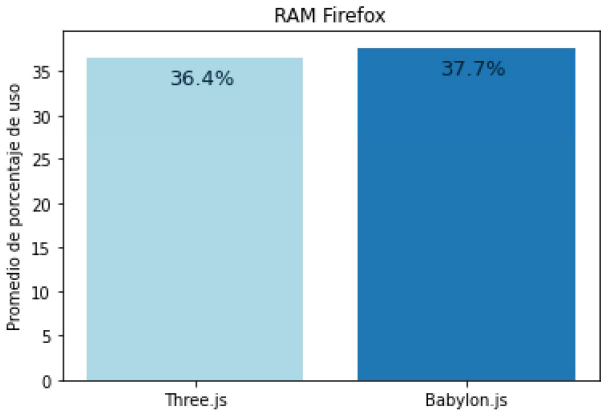


Fig. 40. Promedio de RAM de todos los *benchmarks* pruebas de Firefox.
Fuente: propio del estudio

La Fig. 41 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js en el navegador Brave, con relación al porcentaje de uso de la RAM; donde, con un 1.4% Three.js muestra un mejor desempeño.

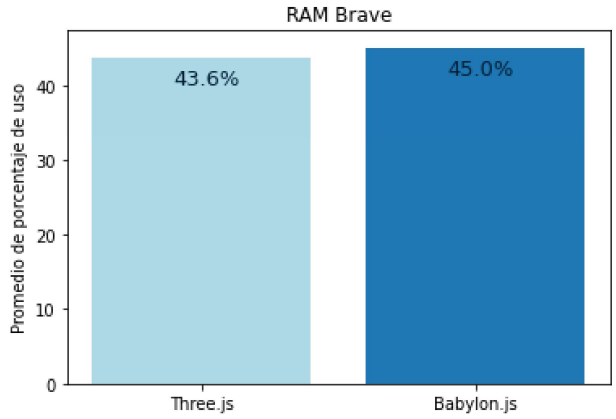


Fig. 41. Promedio de RAM de todos los *benchmarks* pruebas de Brave.
Fuente: propio del estudio

Con un 0.4% menor de consumo en el promedio general de todos los navegadores del porcentaje de uso de RAM: Three.js tiene un mejor rendimiento contra Babylon.js; sin embargo, nuevamente la diferencia es estadísticamente poco significativa, ver Fig. 42.

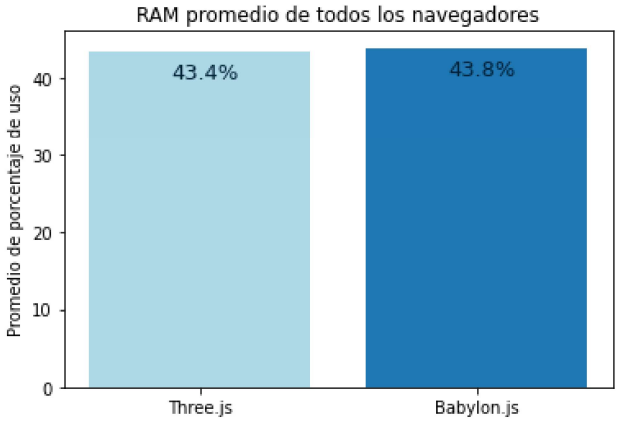


Fig. 42. Promedio de RAM de todos los navegadores.
Fuente: propio del estudio.

2.2.3.4. Resultados del consumo de *Frame Per Second* (FPS)

Los datos presentados en las Tabla 15, Tabla 16 y Tabla 17 son el promedio de todas las medidas realizadas durante las pruebas de rendimiento en relación con los FPS generados durante el tiempo de ejecución. Cabe considerar que, en los siguientes resultados un valor mayor de FPS tiene mejor rendimiento.

En la Tabla 15 se presenta el promedio de los FPS renderizados por el navegador Chrome. Los datos muestran un menor consumo de recursos de la CPU por parte de Babylon.js en todas las pruebas: con una diferencia porcentual que va desde el 2% hasta el 8.5%.

Tabla 15. FPS promedio en Chrome.

Número de iteración	Three.js	Babylon.js
1	67.5	37.2
2	132.2	91.9
3	133.2	101.5

Fuente: propio del estudio.

En la Tabla 16 se presenta el promedio del uso de la FPS en el navegador Firefox. Los datos muestran un menor consumo de recursos de la CPU por parte de Babylon.js en todas las pruebas: con una diferencia porcentual que va desde el 2% hasta el 8.5%.

Tabla 16. FPS promedio en Firefox.

Número de iteración	Three.js	Babylon.js
1	141.8	90.1
2	141.0	90.0
3	138.0	88.9

Fuente: propio del estudio.

En la Tabla 17 se presenta el promedio del uso de los FPS en el navegador Brave. Los datos muestran un menor consumo de recursos de la CPU por parte de Babylon.js en todas las pruebas: con una diferencia porcentual que va desde el 2% hasta el 8.5%.

Tabla 17. FPS promedio en Brave.

Número de iteración	Three.js	Babylon.js
1	146.0	94.0

2	138.9	90.9
3	138.3	114.5

Fuente: propio del estudio.

Este es el único apartado donde se puede ver una clara diferencia entre ambas librerías, ver Fig. 43, Fig. 44 y Fig. 45. Con una mejora de hasta un 36% a favor de Three.js, este tiene un mejor rendimiento.

La Fig. 43 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js, con relación a los FPS renderizados por el navegador Chrome; donde, con 34 FPS más, Three.js muestra un mejor desempeño.

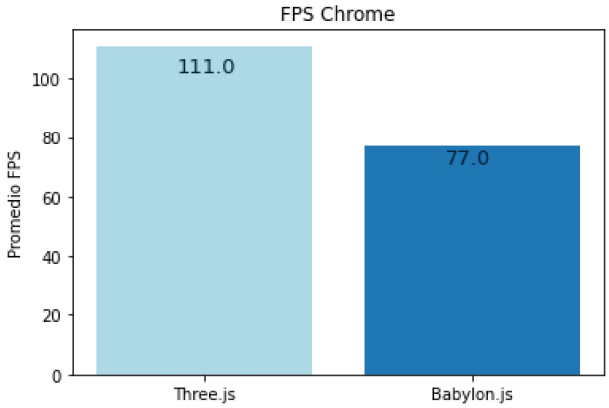


Fig. 43. Promedio de FPS de todos los *benchmarks* pruebas de Chrome.

Fuente: propio del estudio.

La Fig. 44 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js, con relación a los FPS renderizados por el navegador Firefox; donde, con 50.6 FPS más, Three.js muestra un mejor desempeño.

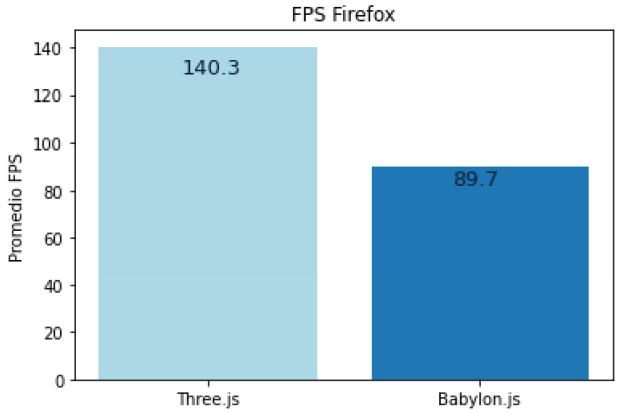


Fig. 44. Promedio de FPS de todos los *benchmarks* pruebas de Firefox.

Fuente: propio del estudio.

La Fig. 45 presenta el promedio general de todas las pruebas de rendimiento entre Three.js y Babylon.js, con relación a los FPS renderizados por el navegador Brave; donde, con 41.3 FPS más, Three.js muestra un mejor desempeño.

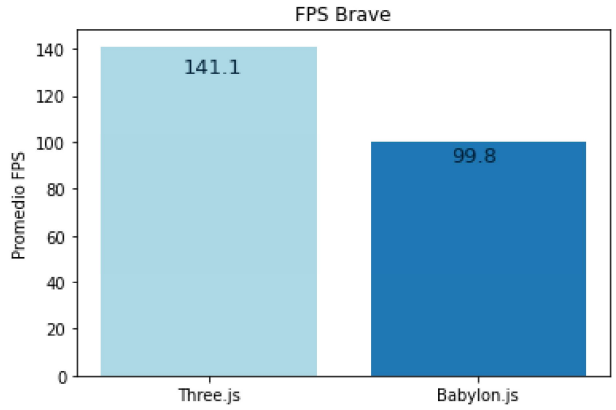


Fig. 45. Promedio de FPS de todos los *benchmarks* pruebas de Brave.

Fuente: propio del estudio.

En el promedio general de todos los navegadores: Three.js tiene 42 FPS más que Babylon.js; eso significa una mejoría de 16.8%, ver Fig. 46. Porcentaje calculado sobre 249 que es el máximo de FPS. Esta es la diferencia más grande entre ambas librerías.

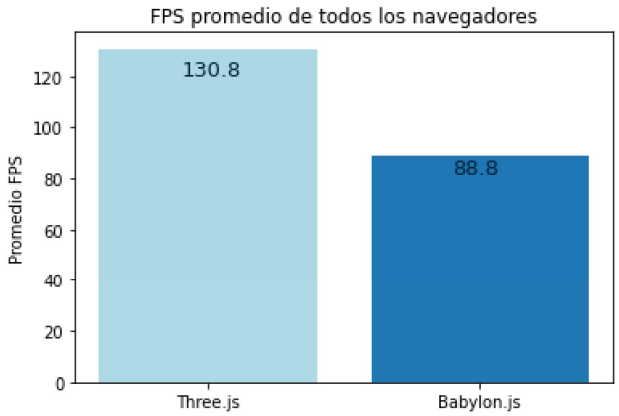


Fig. 46. Promedio de FPS de todos los navegadores.

Fuente: propio del estudio.

2.2.4. Análisis de los resultados

Ambas librerías tienen un rendimiento similar en la mayoría de los apartados: CPU, GPU y RAM. Esto es entendible si se considera que son proyectos especializados en el renderizado 3D para navegadores con varios años de desarrollo.

De modo más detallado, en CPU hay una mejor optimización en Babylon.js con un consumo de hasta un 8.1% menor. Mientras, en GPU y RAM hay una diferencia entre un 0.1% hasta 1.5% a favor de Three.js, mostrando un rendimiento similar.

Sin embargo, en los FPS hay una mejor optimización de recursos a favor de Three.js: con una mejora de hasta un 16.8%, es la métrica con la mayor diferencia entre ambas librerías.

En términos absolutos, Babylon.js solo es mejor que Three.js en CPU, una de cuatro métricas. Aun así, la métrica más importante orientada a la experiencia de usuario son los FPS: porque a menor FPS mayor es el retraso entre la acción del usuario, por ejemplo: hacer clic con el ratón, y lo que ve en pantalla; cuanto mayor es el retraso peor es la experiencia (Ellerweg, 2019; Montagud et al., 2018).

En este aspecto, y debido a la poca diferencia en el resto de las métricas, el factor determinante fueron los FPS. Dejando a Three.js como la mejor librería para desarrollar aplicaciones web con WebGL.

2.3. Selección de librerías de visualización de información

Para la selección de la librería de visualización de información se consideraron dos fases. En la primera fase se aplicaron criterios de inclusión. Los criterios de inclusión

considerados fueron: (i) licencia de código abierto, (ii) JavaScript, iii) soporte para visualizaciones en 3D. Todas las librerías están relacionadas con la categoría de *Data visualization* con contribuciones durante el último año (2019 - 2020) en GitHub.

En la segunda fase se aplicaron criterios relacionados con la relevancia dada por los desarrolladores a cada librería. Las librerías encontradas se ordenaron por el filtro de *Most stars* y se escogieron las dos librerías más relevantes, ver Fig. 47.

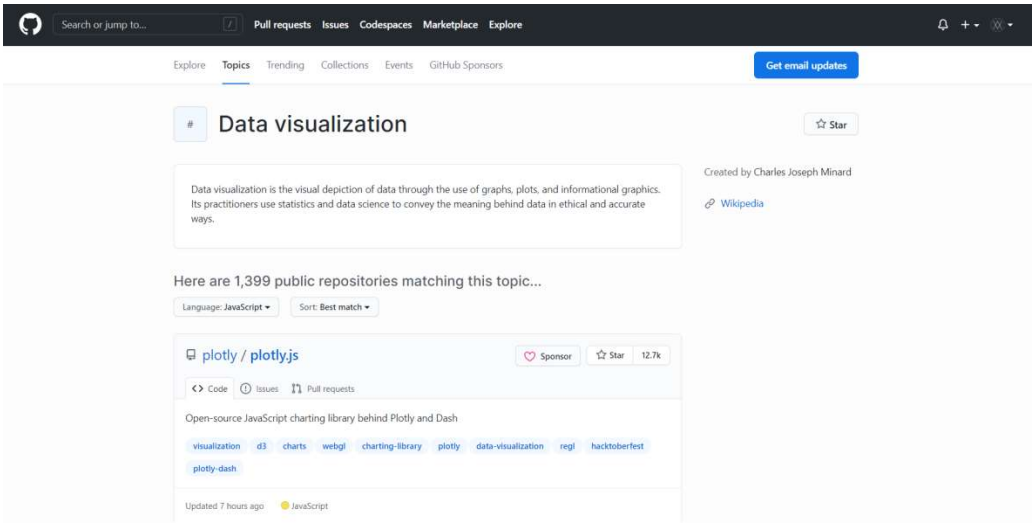


Fig. 47. Librerías de visualización de información seleccionada en la base de datos de GitHub.
Fuente: propio del estudio.

La librería seleccionada se escogió en el mes de diciembre del 2020 en base a datos recopilados hasta esa fecha en GitHub y se presentan en la TABLA 18.

TABLA 18. Librerías de visualización de información seleccionada

Nombre de la librería	Lenguaje de programación	Última *release*	*Stars*	Licencia	Soporte para visualizaciones en 3D
Plotly.js	JavaScript	21 de diciembre de 2020	12.7 k	MIT	Si

Fuente: basado en datos de (plotly, 2020c)

2.4. Diseño del modelo interactivo

La aplicación de visualización de información se desarrolló con la librería resultante de la comparativa de rendimiento para renderizado 3D: Three.js; y, de la búsqueda en la base de datos de GitHub para la visualización de información: Plotly.js.

Adicionalmente, se utilizó tecnologías web para complementar las librerías: HTML, CSS y JavaScript. Una plantilla, antes mencionada, se construyó para el desarrollo de la mayoría de las aplicaciones de este trabajo, incluida esta.

En este trabajo se hace la distinción entre Frontend, la parte que interactúa con el usuario, y Backend, la parte que donde se ejecuta la lógica de negocio; donde, la aplicación construida para este trabajo es exclusivamente el Frontend. Para el Backend se consume una API externa para obtener los datos de la reducción de dimensiones a ser modelados.

El diagrama del proceso de la aplicación, Frontend, se puede ver en la Fig. 2; mientras, un diagrama de bloques de la arquitectura se presenta a continuación, ver Fig. 48.

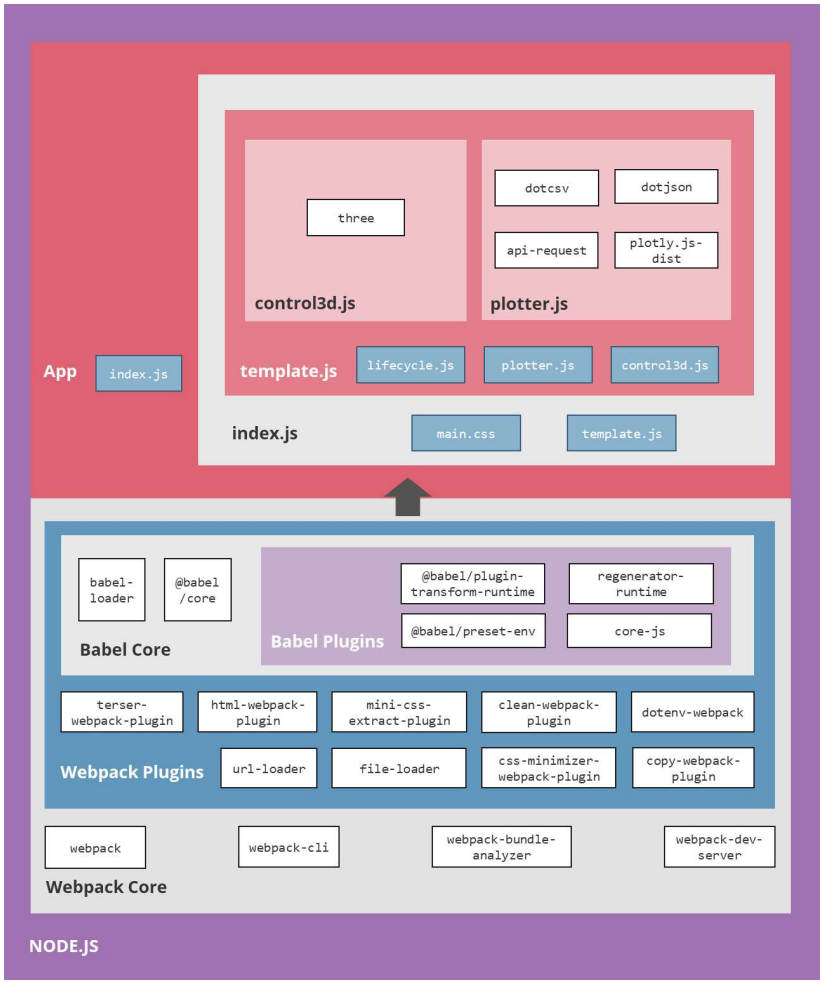


Fig. 48. Diagramas de bloques de la aplicación de visualización de información
Fuente: propio del estudio.

2.4.1. Detección de caras de una figura geométrica

La parte más importante del diseño del modelo interactivo es el algoritmo que reconoce los porcentajes de cada cara mostrada en pantalla.

Dicho algoritmo es genérico y puede ser implementado en cualquier figura geométrica con pequeñas modificaciones. Por ejemplo: en este trabajo se utilizó el dodecaedro, es de doce caras y ayudó a probar el algoritmo en figuras complejas, además de comprobar la usabilidad del esta.

Continuando con el ejemplo, un dodecaedro puede mostrar hasta cinco caras al mismo tiempo y como mínimo tres, estas serían el número de *kernels* que se van a mezclar a la vez. En la Fig. 49 se puede ver cuatro caras al mismo tiempo.

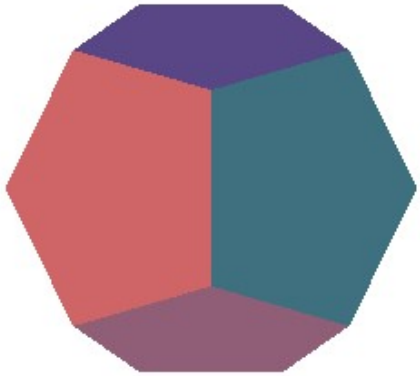


Fig. 49. Dodecaedro mostrando cuatro caras.

Las cuatro caras en conjunto representan el 100%, ver Fig. 50, de lo que se ve en pantalla, distribuyéndose este valor entre las cuatro caras: 30.90% (verde azulado oscuro), 30.90% (salmón), 19.10% (purpura), 19.10% (rosado oscuro).

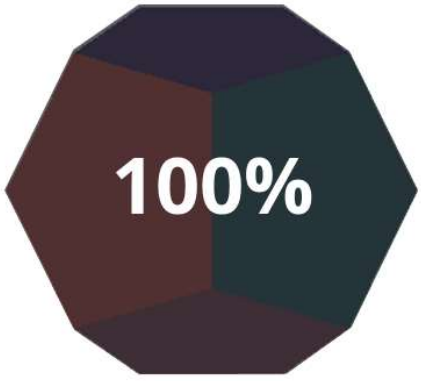


Fig. 50. Las cuatro caras del dodecaedro suman el 100%.

De forma más detallada el algoritmo es el siguiente:

$$\vec{C} - \vec{M} = \vec{D}$$

Donde $\vec{C}$ es el vector posición de la Cámara, $\vec{M}$ es el vector posición del Mesh, o figura geométrica, y $\vec{D}$ es el vector resultante entre la Cámara y el Mesh.

$$Normalize(\vec{\ }) = \vec{P}$$

Este resultado se lo normaliza para que las coordenadas del vector resultante estén entre 0 y 1, a este vector se lo llamará $\vec{P}$.

$$AV(i) = \overrightarrow{N_i} \cdot \vec{P}$$

Donde la función representa un ciclo *for* que calcula el producto punto entre la Normal de cada triángulo, o *shader*, de cara de la figura geométrica, denotada por $\overrightarrow{N_i}$, y el vector $\vec{P}$. Esto dará como resultado una lista de números reales que representan el valor de los ángulos de cada cara respecto a vista de la Cámara.

$$TotalAngle = \sum_{i=1}^{n} AV_i \quad ; AV_i > 0$$

De esta lista de valores solo los valores mayores a 0 son los vectores que apuntan a la Cámara, y son los únicos que se van a tomar en cuenta para calcular los porcentajes de cada cara visible de la figura geométrica.

$$AngleOverTotalAngle(AV_i) = \frac{AV(i)}{TotalAngle}$$

A continuación, se divide el $TotalAngle$ sobre cada valor de la lista $AV(i)$ para calcular la fracción que equivale cada cara en relación con el total de las caras visibles, este valor se calcula sobre 1.

Como anotación: dependiendo de la figura geométrica que se use esta parte cambiará. En este caso se utilizó un Dodecaedro que tiene 12 caras y por cada una se renderiza 3 triángulos o *shaders*, lo que da un total de 36 triángulos.

$$FacePercentage = \sum_{i=1}^{n} AngleOverTotalAngle(AV_i)$$

Ahora, tal como se dijo, esto cambiará con relación a la figura geométrica utilizada. Para el Dodecaedro se suma cada tres triángulos, dando la fracción del total que representa esa cara de todas las visibles en pantalla.

Este valor es el que se envía a la API externa para que realice el cálculo de la Reducción de Dimensiones. Para mostrarse en la interfaz del usuario se lo multiplica por 100 para convertirlo a porcentaje.

2.4.2. Diseño web adaptable

La forma de consumir contenido cambia entre distintos dispositivos, razón por la cual se desarrolló una interfaz adaptable a distintos tipos de pantallas. En enfoque estuvo principalmente entre teléfonos inteligentes y de escritorio.

La interfaz móvil se la diseño en dos partes, ver Fig. 51: en la parte superior de la pantalla, para la visualización de la información; y, en la parte inferior, la interacción con el modelo interactivo.

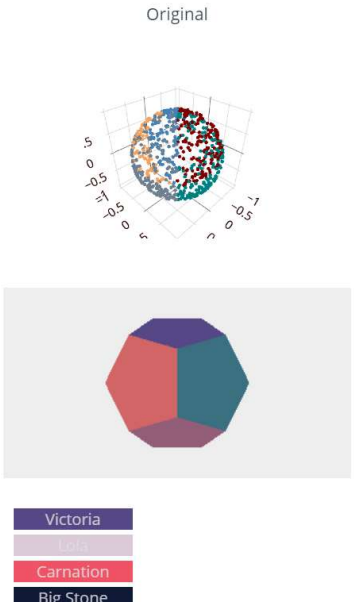


Fig. 51. Versión móvil del modelo interactivo.
Fuente: propio del estudio.

La versión de escritorio, ver Fig. 52, también está diseñada en dos partes: dos tercios izquierdos de la pantalla, para la visualización de la información; y, el tercio derecho restante, para la interacción con el modelo interactivo.

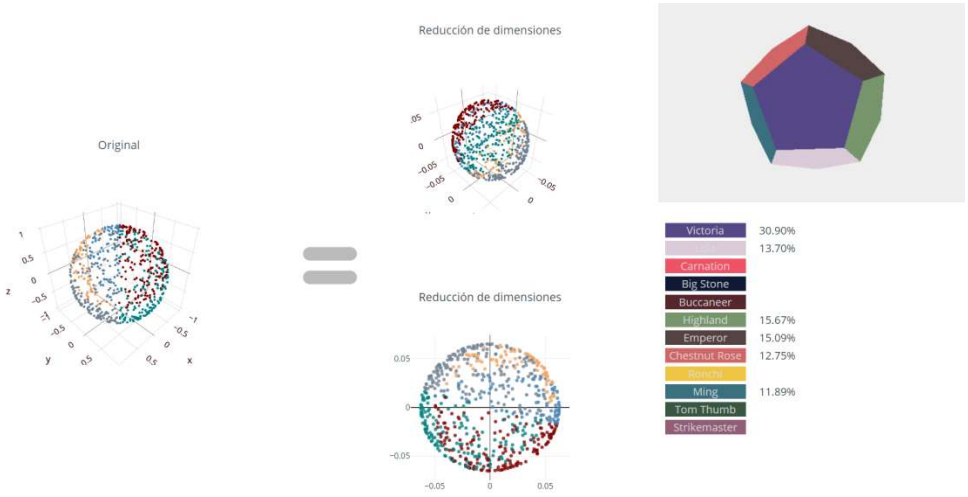


Fig. 52. Versión de escritorio del modelo interactivo.
Fuente: propio del estudio.

Dado que el modelo interactivo está pensado para personas no expertas en reducción de dimensiones, se vio necesaria la introducción de un manual de uso antes de empezar a interactuar con este, ver Fig. 53; con el objetivo de realizar un preámbulo de los conceptos de RD y forma de uso del modelo.

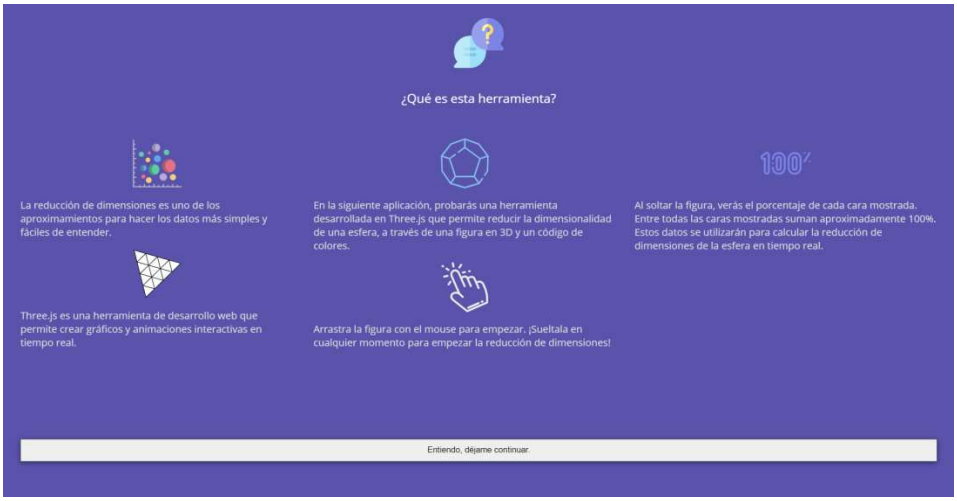


Fig. 53. Versión de escritorio de manual de uso del modelo interactivo.
Fuente: propio del estudio.

Una versión para teléfonos inteligentes de esta introducción también fue desarrollada con la intención de abarcar la mayor cantidad de usuarios, ver Fig. 54.

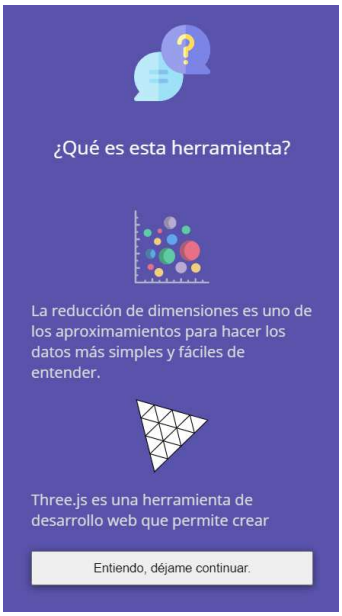


Fig. 54. Versión de móvil de manual de uso del modelo interactivo.

Fuente: propio del estudio.

CAPÍTULO 3

Validación de resultados

La Escala de Usabilidad del Sistema, o *System Usability Scale* (SUS) en inglés, es un cuestionario de 10 preguntas que se responde con una escala de Likert de cinco niveles que van de: “Totalmente en desacuerdo” a “Totalmente de acuerdo” (Hedlefs Aguilar et al., 2016; Lewis, 2018).

SUS, publicado por (Brooke, 1996) es uno de los cuestionarios más utilizados para medir la percepción de la usabilidad. Su interpretación es de carácter unidimensional, es decir, mide de forma general la percepción de la usabilidad (Lewis, 2018).

3.1. Metodología

El cuestionario de SUS original consta de cinco preguntas positivas y cinco negativas, posteriormente se desarrolló una versión alternativa con solo preguntas positivas (Hedlefs Aguilar & Garza Villegas, 2016).

En este trabajo para la evaluación de la percepción de la usabilidad del “Modelo interactivo de visualización de información utilizando una librería de renderizado 3D en aplicaciones web aplicado a la reducción de dimensiones” se utilizó la adaptación de este cuestionario al español de (Hedlefs Aguilar & Garza Villegas, 2016) en su versión positiva, tal como recomiendan los autores de esta adaptación.

El cuestionario consta de las siguientes 10 preguntas:

1. Creo que me gustaría utilizar frecuentemente este sitio web.
2. Encontré el sitio web sencillo.
3. Pienso que el sitio web es fácil de usar.
4. Pienso que podré utilizar este sitio web sin el apoyo de personal técnico.
5. Encontré que varias de las funciones en el sitio en el sitio web estaban bien integradas.
6. Pensé que había demasiada consistencia en el sitio web.
7. Me imagino que la mayoría de las personas podrían aprender a usar este sitio web muy rápido.
8. Encontré el sitio web muy intuitivo.
9. Me sentí muy confiado (seguro) al utilizar el sitio web.
10. Pude utilizar el sitio web sin tener que aprender nada nuevo.

Estas se evalúan con una escala de Likert de cinco niveles:

1. Totalmente en desacuerdo.
2. En desacuerdo.
3. Ni de acuerdo, ni en desacuerdo.
4. De acuerdo.
5. Totalmente de acuerdo.

Tal como se mencionó antes, con el cuestionario SUS se puede evaluar la característica de usabilidad de la ISO/IEC 25010 de forma unidimensional, es decir, de forma general toda la característica (Wulandari & Aristana, 2021). Razón por la que se tomó los resultados de la encuesta para evaluar la característica de usabilidad de la ISO/IEC 25010.

Dicha encuesta fue efectuada a través de la plataforma de Office 365, ver Anexo B: Encuesta de usabilidad SUS, a estudiantes de ingeniería de Software de diferentes niveles. Además, con el fin de que los estudiantes puedan probar el modelo interactivo, este se sirvió en Netlify: una plataforma en la nube que se utiliza para desplegar aplicaciones Frontend (Netlify, 2021). Para el Backend se consume una API externa para obtener los datos de la reducción de dimensiones a ser modelados.

Todo el proceso se lo realizó de forma remota, a través de Microsoft Teams. Con un total de 34 personas encuestadas, los resultados se muestran en la Tabla 19.

Tabla 19. Resultados de la encuesta

Preguntas	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Pregunta 1	0	1	8	18	7
Pregunta 2	0	0	2	12	20
Pregunta 3	0	0	3	15	16
Pregunta 4	0	1	4	20	9
Pregunta 5	0	1	2	15	16
Pregunta 6	1	2	8	15	8
Pregunta 7	0	1	4	14	15
Pregunta 8	0	0	2	13	19

Pregunta 9	0	0	3	15	16
Pregunta 10	0	2	3	18	11

Fuente: propio del estudio.

3.2. Análisis de resultados

A continuación, se detallan los resultados obtenidos de cada pregunta de la encuesta SUS aplicada a los estudiantes de la carrera de Software.

Pregunta 1: Creo que me gustaría utilizar frecuentemente este sitio web.

La Fig. 55 corresponde a la pregunta 1 y en su análisis se puede observar que la opción más votada fue “De acuerdo” con un 54% de los votos, mientras la segunda opción fue “Totalmente de acuerdo” con 26% de los votos. Esto muestra que, aunque hay predisposición de los usuarios en volver a utilizar el sitio web, todavía hay rango de mejora para obtener mejores valoraciones.

Fig. 55. Resultados de la pregunta 1 de la encuesta SUS.

Pregunta 2: Encontré el sitio web sencillo.

La Fig. 56 corresponde a la pregunta 2 y en su análisis se puede observar que la opción más votada fue “Totalmente de acuerdo” con un 65% de los votos, la segunda mejor opción fue “De acuerdo” con un 31% de los votos. Denotando que los usuarios no

encuentran la interfaz de usuario compleja, a tal punto, que puedan confundirlos de su tarea.

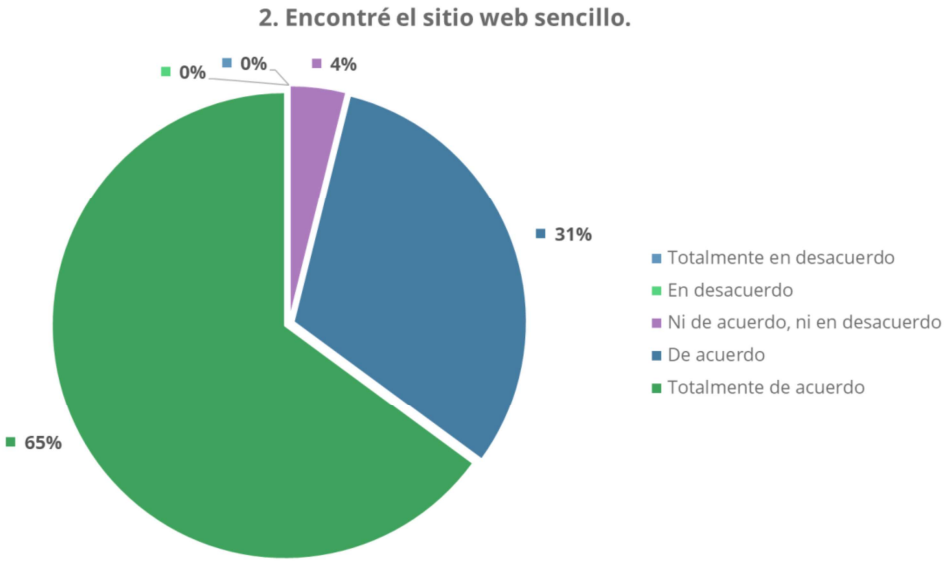


Fig. 56. Resultados de la pregunta 2 de la encuesta SUS.

Pregunta 3: Pienso que el sitio web es fácil de usar.

La Fig. 57 corresponde a la pregunta 3 y en su análisis se puede observar que la opción más votada fue “Totalmente de acuerdo” con un 54% de los votos, como segundo lugar, “De acuerdo” acumula el 40% de los votos. Esta pregunta valida que el modelo interactivo es usable incluso por personas no expertas, que no sintieron que el dodecaedro sea complicado de utilizar.

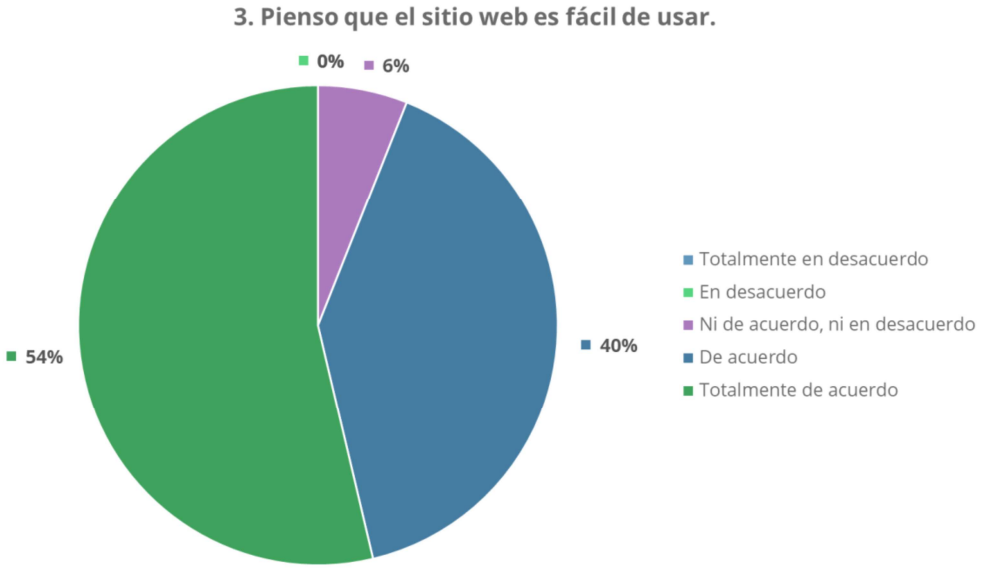


Fig. 57. Resultados de la pregunta 3 de la encuesta SUS.

Pregunta 4: Pienso que podré utilizar este sitio web sin el apoyo de personal técnico.

La Fig. 58 corresponde a la pregunta 4 y en su análisis se puede observar que la opción más votada fue “De acuerdo” con un 58% de los votos, seguido de “Totalmente de acuerdo” con el 32% de los votos. Los resultados muestran que los usuarios necesitan poca o ninguna ayuda para hacer uso del modelo interactivo del sitio web. Una nota buena, considerando que son usuarios no expertos.

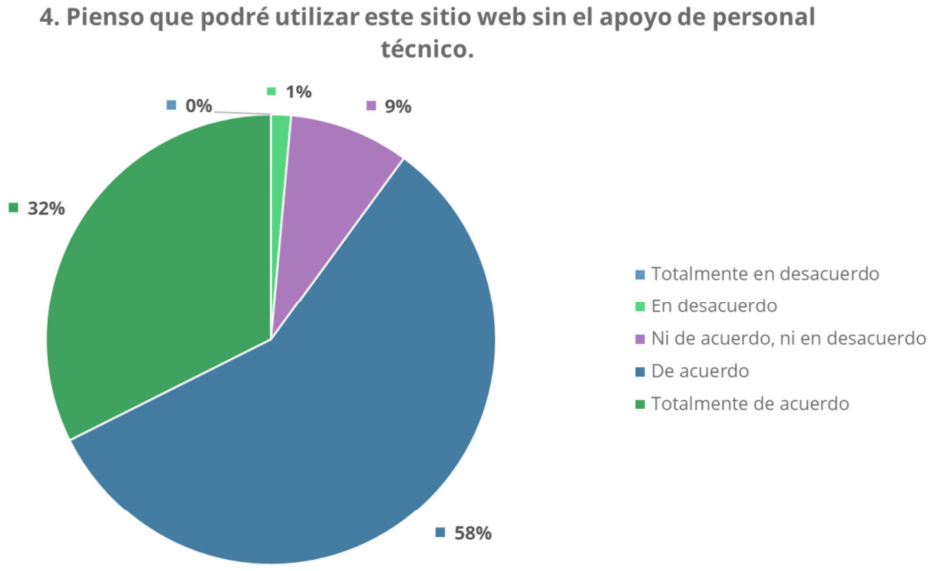


Fig. 58. Resultados de la pregunta 4 de la encuesta SUS.

Pregunta 5: Encontré que varias de las funciones en el sitio en el sitio web estaban bien integradas.

La Fig. 59 corresponde a la pregunta 5 y en su análisis se puede observar que la opción más votada fue “Totalmente de acuerdo” con un 54% de los votos, la segunda mejor opción fue “De acuerdo” con un 41% de los votos. Lo que muestra que las funciones del modelo interactivo son funcionales y no presentan problemas entre ellas.

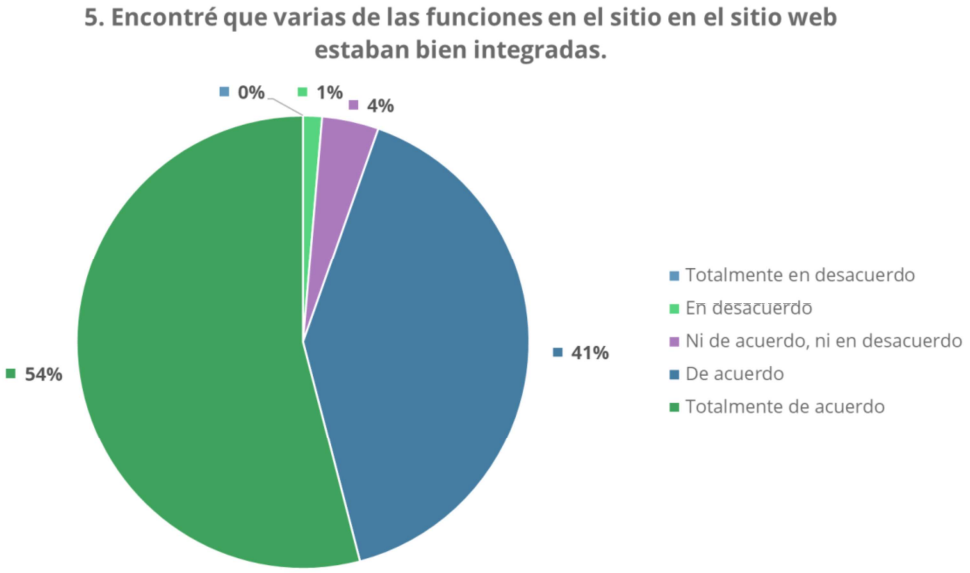


Fig. 59. Resultados de la pregunta 5 de la encuesta SUS.

Pregunta 6: Pensé que había demasiada consistencia en el sitio web.

La Fig. 60 corresponde a la pregunta 6 y en su análisis se puede observar que la opción más votada fue “De acuerdo” con un 46% de los votos, seguido del 31% con la opción “Totalmente de acuerdo”. Los resultados muestran que la consistencia del sitio es correcta.

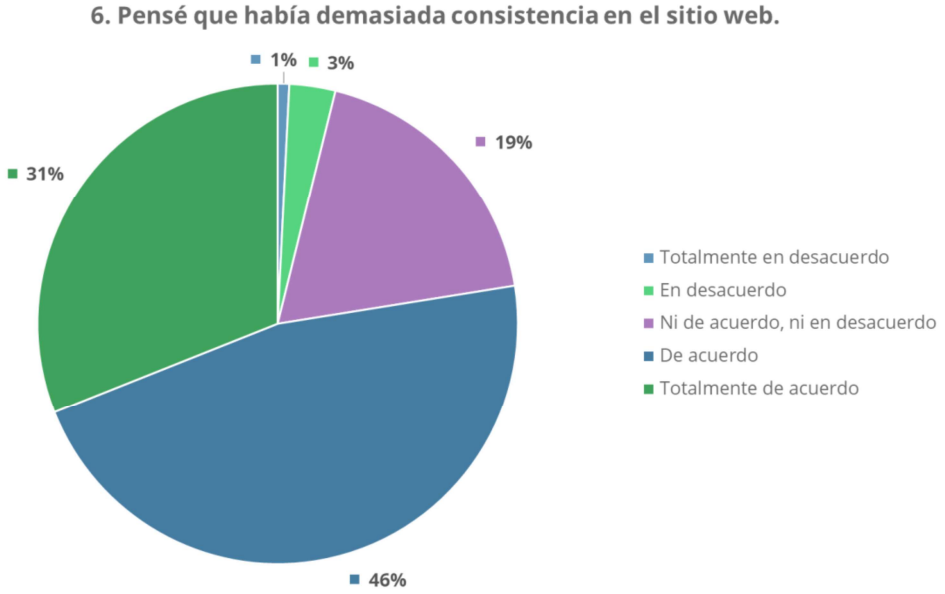


Fig. 60. Resultados de la pregunta 6 de la encuesta SUS.

Pregunta 7: Me imagino que la mayoría de las personas podrían aprender a usar este sitio web muy rápido.

La Fig. 61 corresponde a la pregunta 7 y en su análisis se puede observar que la opción más votada fue “Totalmente de acuerdo” con un 52% de los votos, el segundo lugar fue “De acuerdo” con el 39% de los votos. Esto muestra que el sitio web es ideal para personas no expertas en mezcla de reducción de dimensiones, ya que no necesitan de una alta curva de aprendizaje para utilizarlo.

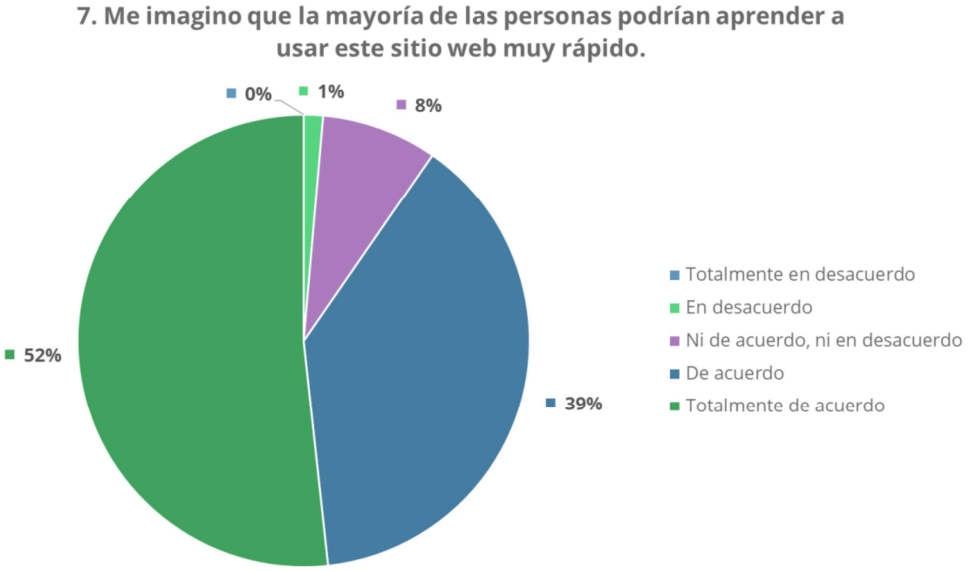


Fig. 61. Resultados de la pregunta 7 de la encuesta SUS.

Pregunta 8: Encontré el sitio web muy intuitivo.

La Fig. 62 corresponde a la pregunta 8 y en su análisis se puede observar que la opción más votada fue “Totalmente de acuerdo” con un 62% de los votos, seguido del 34% con la opción “De acuerdo”. Según los datos mostrados, el usuario muestra una percepción clara de lo que tiene que hacer solo manipulando el modelo interactivo, sin necesidad de conocimiento profundos del tema.

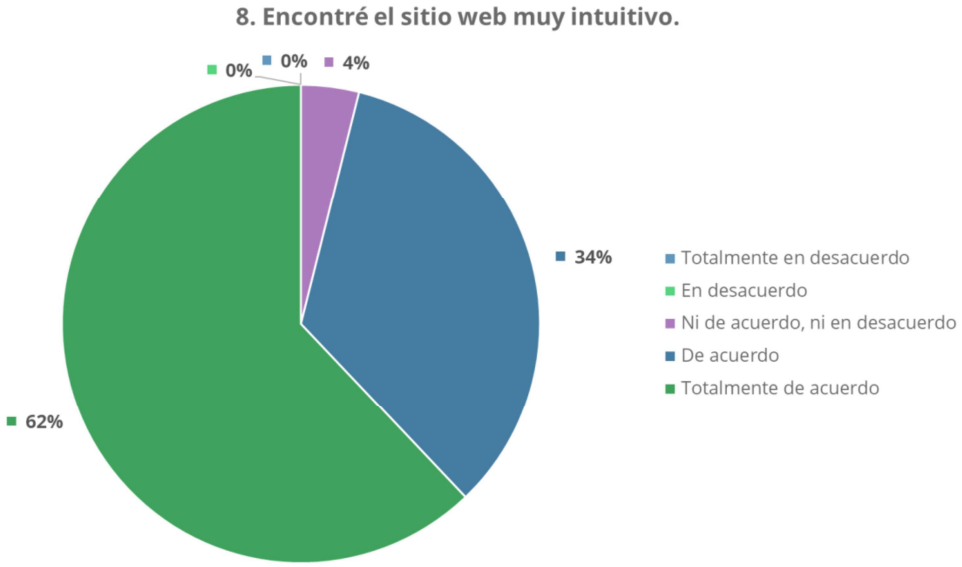


Fig. 62. Resultados de la pregunta 8 de la encuesta SUS.

Pregunta 9: Me sentí muy confiado (seguro) al utilizar el sitio web.

La Fig. 63 corresponde a la pregunta 9 y en su análisis se puede observar que la opción más votada fue “Totalmente de acuerdo” con un 54% de los votos, mientras la segunda opción fue “De acuerdo” con un 40% de los votos. Según los resultados, el usuario muestra confianza en que no va a haber errores en el sitio web que afecten su experiencia.

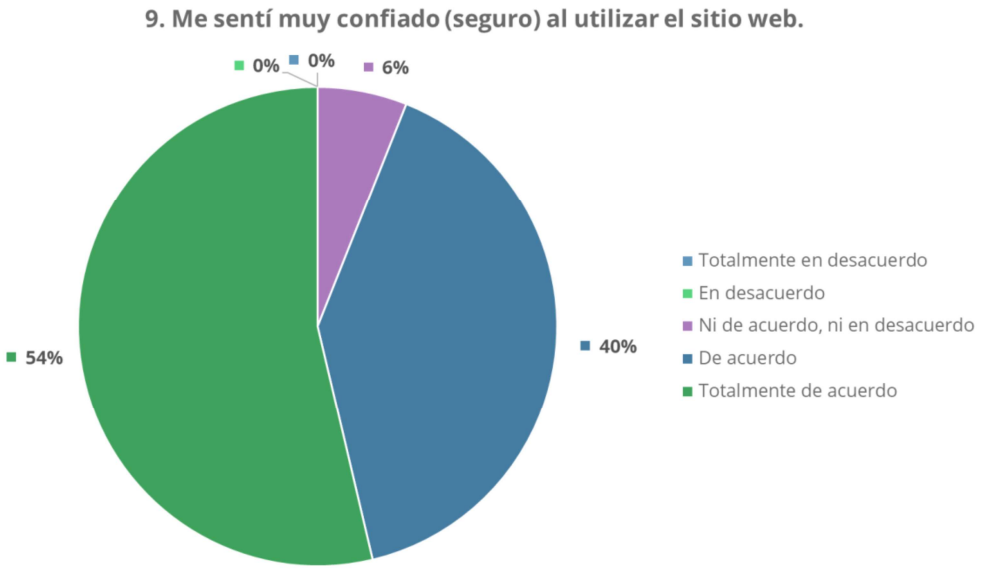


Fig. 63. Resultados de la pregunta 9 de la encuesta SUS.

Pregunta 10: Pude utilizar el sitio web sin tener que aprender nada nuevo.

La Fig. 64 corresponde a la pregunta 10 y en su análisis se puede observar que la opción más votada fue “De acuerdo” con un 52% de los votos, seguido del 39% por la opción “Totalmente de acuerdo”. El modelo interactivo permite realizar la mezcla de reducción de dimensiones sin muchos conocimientos; sin embargo, todavía es necesario cierto entendimiento básico antes de empezar a utilizarlo.

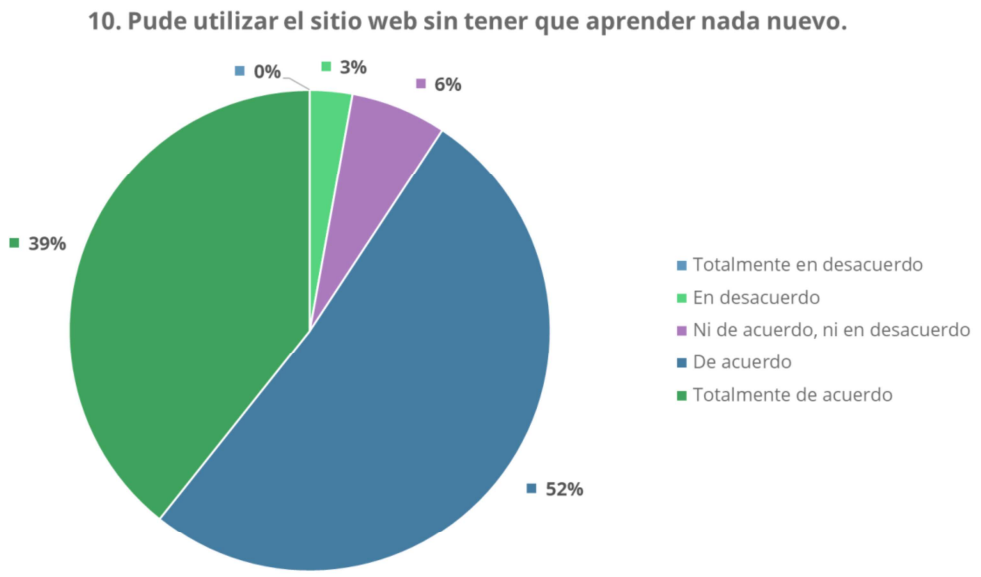


Fig. 64. Resultados de la pregunta 10 de la encuesta SUS.

Finalmente, en la Fig. 65, se muestra un resumen de los resultados de la encuesta SUS usada para evaluar la usabilidad del modelo interactivo. En esta se puede observar como la aceptación general demuestra que el modelo interactivo ha cumplido con su objetivo de ayudar a personas no expertas, y también a expertas, a realizar la exploración de mezcla de métodos de reducción de dimensiones de forma fácil y amena.

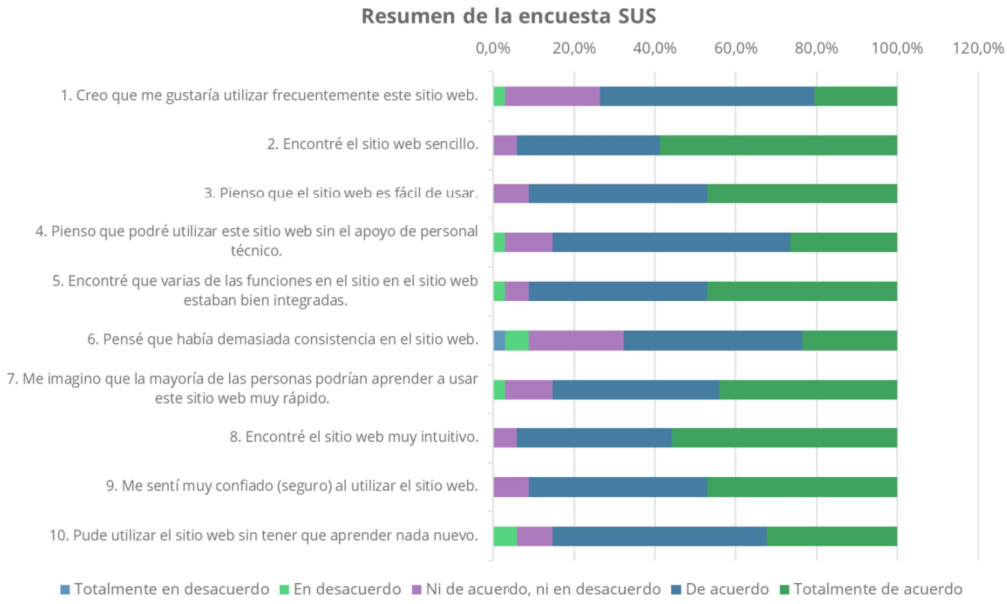


Fig. 65. Resumen de la encuesta SUS aplicada al modelo interactivo.

3.3. Interpretación de resultados

Para obtener un resultado concluyente de la encuesta SUS, hay que realizar un análisis final que dará un único resultado. Este análisis se lo realizó en base al trabajo de (Hedlefs Aguilar & Garza Villegas, 2016) que realizó una comparativa entre las versiones positivas y negativa de la encuesta.

Primero, para realizar el análisis hay que multiplicar el resultado de cada pregunta, ver Tabla 19, por un cierto valor, que va desde: 1 para las respuestas de "Totalmente en desacuerdo" hasta 5 para las respuestas de "Totalmente de acuerdo".

Por ejemplo: la opción "En desacuerdo" de la pregunta 1 tiene el valor de 1, este número se lo multiplicará por 2 lo que dará como resultado 2 (1 * 2 = 2); este será el nuevo valor que utilizará para hacer el análisis.

En la Tabla 20 se puede ver los resultados obtenidos:

Tabla 20. Resultados de la normalización de las preguntas

Preguntas	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuerdo	Totalmente de acuerdo
Pregunta 1	0	2	24	72	35
Pregunta 2	0	0	6	48	100
Pregunta 3	0	0	9	60	80
Pregunta 4	0	2	12	80	45
Pregunta 5	0	2	6	60	80
Pregunta 6	1	4	24	60	40
Pregunta 7	0	2	12	56	75
Pregunta 8	0	0	6	52	95
Pregunta 9	0	0	9	60	80
Pregunta 10	0	4	9	72	55

Fuente: propio del estudio.

Segundo, se tomará el valor resultante del promedio de cada pregunta y se le restará 1. Por ejemplo, al promedio de la pregunta 1 que es 3,91 se le resta 1, lo que da como resultado: 2,91 (3,91 – 1 = 2,91).

Una vez realizado los promedios correspondientes de cada una de las preguntas se procede a sumar todos los valores obtenidos, de esta manera se obtiene el promedio total.

En la Tabla 21 se puede ver los resultados obtenidos:

Tabla 21. Resultados de las preguntas

Preguntas	**Resultado**	**Promedio** (Resultado - 1)
Pregunta 1	3,91	2,91
Pregunta 2	4,53	3,53
Pregunta 3	4,38	3,38
Pregunta 4	4,09	3,09
Pregunta 5	4,35	3,35
Pregunta 6	3,79	2,79
Pregunta 7	4,26	3,26
Pregunta 8	4,50	3,50
Pregunta 9	4,38	3,38
Suma total		32,32

Fuente: propio del estudio.

Finalmente, para obtener el resultado final se suman los promedios totales y dicho valor se multiplica por 2,5. En este caso, la suma total es 32,32 que multiplicado por 2,5 da **80,81** (32,32 * 2,5 = 80,81).

De esta manera se obtiene el resultado final. Una vez obtenido el resultado final se puede determinar la calificación respectiva del modelo interactivo.

Y, para este fin, se utilizará una escala propuesta por (Sauro, 2018) para interpretar los resultados de la encuesta SUS, ver Fig. 66. Esta denota cinco formas de interpretar los resultados: Percentiles (*SUS Score*); Grados (*Grade*); Adjetivos (*Adjective*); Aceptabilidad (*Acceptable*); y, Promotores y detractores (*NPS*), las cuales son perfectamente compatibles y se superponen entre ellas.

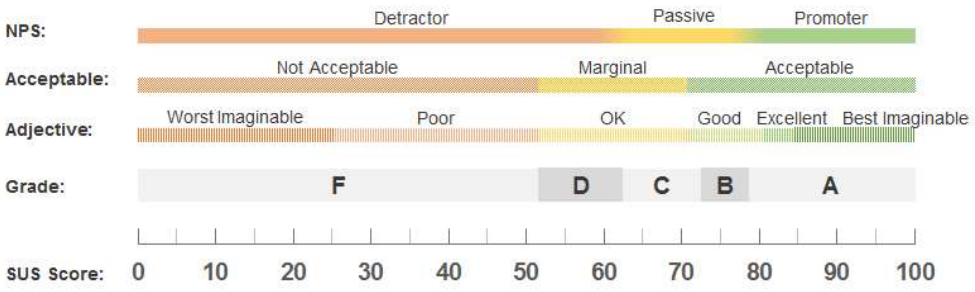


Fig. 66. Grado de calificación SUS.

Fuente: (Sauro, 2018)

Según (Sauro, 2018), se puede utilizar cualquier interpretación, o incluso todas al mismo tiempo. En este trabajo, se utilizó Grados y Adjetivos como una sola interpretación de los resultados.

De tal forma que, al tener un *SUS Score* de 80,81 esto equivale a **Grado A** en la escala de (Sauro, 2018), lo que significa que tiene un rendimiento superior a la media; además, con este puntaje se le califica con el Adjetivo **Excelente**, que representa la experiencia del usuario, y en este caso significa que el modelo interactivo todavía puede seguirse mejorando.

Por último, la relación entre la encuesta SUS y la característica de Usabilidad de la ISO 25010 es directa y general (Lewis, 2018; Wulandari & Aristana, 2021). Por lo tanto, al tener el "Modelo interactivo de visualización de información utilizando librerías de renderizado 3D en aplicaciones web, aplicado a la reducción de dimensiones" una calificación de 80,81, con Grado A y con el Adjetivo de Excelente, se considera que cumple con la característica de usabilidad.

CONCLUSIONES Y RECOMENDACIONES

Conclusiones:

En el ámbito del desarrollo de aplicaciones 3D con Three.js y Babylon.js: hay librerías de terceros que integran nuevas características como el motor de físicas en tiempo real. En este aspecto, Babylon.js tiene una mejor integración nativa que Three.js.

No existe una integración oficial de Ammo.js, motor de físicas en tiempo real, con Three.js; mientras, si la hay con Babylon.js.

En consumo de CPU: Babylon.js consume hasta un 8.1% menos que Three.js, siendo la métrica en la que destaca más a su favor.

A nivel de software: Three.js es la mejor librería, con un 16.8% más FPS renderizados en las pruebas.

En la comparativa de rendimiento se observó una diferencia en consumo de recursos desde 0.1% hasta el 8.1% a nivel de hardware entre Three.js y Babylon.js, sin claro ganador. Sin embargo, a nivel de manejo de esos recursos, o software: Three.js es superior a Babylon.js por el 16.8%.

Tanto Three.js como Babylon.js tienen una extensa documentación y comunidad de desarrolladores activa.

La encuesta SUS dio como resultado Grado A y Adjetivo Excelente para el modelo interactivo, cumpliendo con la característica de Usabilidad de la ISO 25010. Lo que demuestra que es capaz de simplificar el proceso de la mezcla de reducción de dimensiones al punto de que usuarios no expertos sean capaces de realizarlo.

Recomendaciones:

Si se busca desarrollar aplicaciones 3D con motores de físicas en tiempo real, usar Babylon.js es una mejor opción por defecto.

Enable3D es un *framework* desarrollado sobre Three.js junto con Ammo.js, y es una buena opción para utilizar Three.js con un motor de físicas en tiempo real. Con respecto a Babylon.js, usar la librería nativa es más que suficiente.

Si se necesita desarrollar una aplicación 3D donde reducir el consumo de CPU sea prioritario: Babylon.js es la mejor opción.

Si se necesita desarrollar una aplicación que no sea exigente a nivel de hardware, o su optimización no sea prioritaria, es mejor usar Three.js como librería de renderizado 3D.

Incluso con las diferencias a nivel de Software y Hardware entre ambas librerías de renderizado 3D. Three.js es la mejor opción por defecto, debido a la estabilidad que ofrece, orientado a la experiencia del usuario, y solo se recomendaría usar Babylon.js en ambientes donde sea prioritario optimizar al mínimo detalle el consumo de recursos del hardware, por ejemplo: aplicaciones móviles.

Se recomienda seguir explorando este aproximamiento de modelo interactivo para la mezcla de reducción de dimensiones,

El modelo interactivo demostró ser efectivo que para que usuarios no expertos sean capaces de realizar mezclas de reducción de dimensiones; sin embargo, aún fue necesario cierta explicación adicional antes de empezar a utilizarlo. Este este aspecto, un tutorial guiado puede ser un mejor aproximamiento y se recomienda su uso en lugar de texto antes de dejar utilizar al usuario el modelo interactivo.

Trabajo futuro:

Las pruebas de rendimiento realizadas en este trabajo fueron hechas en una arquitectura de escritorio, una laptop específicamente, con el sistema operativo Windows 10; sin embargo, como trabajo futuro se recomienda probar distintas arquitecturas y sistemas operativos, especialmente en arquitecturas móviles donde minimizar el consumo de los recursos del hardware es prioritario.

La encuesta SUS ha demostrado que el modelo interactivo propuesto es una alternativa viable para explorar la mezcla de reducción de dimensiones, teniendo gran aceptación entre los usuarios. Desde este punto, se recomienda como trabajo futuro seguir explorando este aproximamiento tanto para usuarios expertos, como no expertos en Reducción de Dimensiones.

El modelo de rendimiento fue desarrollado con la figura geométrica de un Dodecaedro (12 caras); sin embargo, los algoritmos utilizados funcionan con otras figuras, como: el Tetraedro (4 caras), Octaedro (8 caras), Icosaedro (20 caras), entre otras. Como trabajo futuro, se recomienda probar con otras figuras, como las mencionadas antes, con el fin de saber cuál es la que da mejor experiencia de usuario.

REFERENCIAS

Agarwal, S. (2014). Data mining: Data mining concepts and techniques. In *Proceedings - 2013 International Conference on Machine Intelligence Research and Advancement, ICMIRA 2013* (pp. 203–207). https://doi.org/10.1109/ICMIRA.2013.45

Alonso, S., Prada, M. A., Fuertes, J. J., Díaz, I., & Domínguez, M. (2015). *Analysis of Electricity Bill Data using Interactive Dimensionality Reduction*. 1–7. https://doi.org/10.1145/2797143.2797163

Angel, E. (2017). The Case for Teaching Computer Graphics with WebGL: A 25-Year Perspective. *IEEE Computer Graphics and Applications*, *37*(2), 106–112. https://doi.org/10.1109/MCG.2017.26

Atlassian Agile Coach. (2020). *Kanban*. Atlassian Agile Coach. https://www.atlassian.com/es/agile/kanban

BabylonJS. (2020a). *Babylon.js*. Github. https://github.com/BabylonJS/Babylon.js

BabylonJS. (2020b). *gITF Loader Demo*. https://playground.babylonjs.com/#WGZLGJ

BabylonJS. (2020c). *Marble Tower ammo.js Demo*. https://playground.babylonjs.com/#3I55DK#0

Bostock, M. (2019). *Parallel Coordinates / D3 / Observable*. Observable. https://observablehq.com/@d3/parallel-coordinates

Brooke, J. (1996). SUS: A “Quick and Dirty” Usability Scale. *Usability Evaluation In Industry*, 207–212. https://doi.org/10.1201/9781498710411-35

Catuhe, D. (2015). WebGL Engine Design in Babylon.js. *WebGL Insights*, 141–156. https://doi.org/10.1201/b18564-14

D3. (2020). *D3.js introduction*. D3.Js. https://d3js.org#introduction

Desjardins, J. (2019). *What Happens in an Internet Minute in 2019?* Visual Capitalist. https://www.visualcapitalist.com/what-happens-in-an-internet-minute-in-2019/

Diego, J. A. P. D. . D. F. A. C. Z. C. E. J., & G., C. (2018). A Novel Color-Based Data Visualization Approach Using a Circular Interaction Model and Dimensionality Reduction. *International Symposium on Neural Networks*, *1*, 407–414. https://doi.org/10.1007/978-3-319-92537-0

ECMA-International. (2020). *Standard ECMA-262*. ECMA-International. https://www.ecma-international.org/publications/standards/Ecma-262.htm

Ellerweg, R. (2019). Make Frame Rate Studies Useful for System Designers. *Proceedings - ICGI 2018: International Conference on Graphics and Interaction*, 1–8. https://doi.org/10.1109/ITCGI.2018.8602965

Gayour, F., & Cantor, D. (2018). *Real-time 3D Graphics with WebGL 2*. Packt Publishing.

Hedlefs Aguilar, M. I., De la Garza González, A., Sánchez Miranda, M. P., & Garza Villegas, A. A. (2016). Adaptación al español del Cuestionario de Usabilidad de Sistemas Informáticos CSUQ / Spanish language adaptation of the Computer Systems Usability Questionnaire CSUQ. *RECI Revista Iberoamericana de Las Ciencias Computacionales e Informática*, *4*(8), 84. https://doi.org/10.23913/reci.v4i8.35

Hedlefs Aguilar, M. I., & Garza Villegas, A. A. (2016). Análisis comparativo de la Escala de Usabilidad del Sistema (EUS) en dos versiones. *RECI Revista Iberoamericana de Las Ciencias Computacionales e Informática*, *5*(10), 44.

ISO/IEC. (2011). *ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733

John D. Kelleher, B. T. (2018). Data Science. In *The MIT Press Essential Knowledge Series*. https://ejournal.poltektegal.ac.id/index.php/siklus/article/view/298%0Ahttp://repositorio.unan.edu.ni/2986/1/5624.pdf%0Ahttp://dx.doi.org/10.1016/j.jana.2015.10.005%0Ahttp://www.biomedcentral.com/1471-2458/12/58%0Ahttp://ovidsp.ovid.com/ovidweb.cgi?T=JS&P

Keim, D. A. (2002). *Information Visualization and Visual Data Mining*. *8*(1), 1–8. http://arxiv.org/abs/1307.8430

Keim, D. A., & Kriegel, H. P. (1996). Visualization techniques for mining large databases: A comparison. *IEEE Transactions on Knowledge and Data Engineering*, *8*(6), 923–938. https://doi.org/10.1109/69.553159

Laksono, D. (2018). Testing Spatial Data Deliverance in SQL and NoSQL Database Using NodeJS Fullstack Web App. *Proceedings - 2018 4th International Conference*

on Science and Technology, ICST 2018, *1*, 1–5. https://doi.org/10.1109/ICSTC.2018.8528705

Lewis, J. R. (2018). Measuring Perceived Usability: The CSUQ, SUS, and UMUX. *International Journal of Human-Computer Interaction*, *34*(12), 1148–1156. https://doi.org/10.1080/10447318.2017.1418805

Li, L., Qiao, X., Lu, Q., Ren, P., & Lin, R. (2020). Rendering Optimization for Mobile Web 3D Based on Animation Data Separation and On-Demand Loading. *IEEE Access*, *8*, 88474–88486. https://doi.org/10.1109/ACCESS.2020.2993613

Mirada. (2020). *Three.js Fundamentals*. Three.Js Fundamentals. https://threejsfundamentals.org/threejs/lessons/threejs-fundamentals.html

Montagud, M., Cesar, P., Boronat, F., & Jansen, J. (2018). MediaSync: Handbook on multimedia synchronization. In *MediaSync: Handbook on Multimedia Synchronization*. https://doi.org/10.1007/978-3-319-65840-7

Moreau-Mathis, J. (2016). *Babylon.js Essentials : understand, train, and be ready to develop 3D web applications/video games using Babylon.js framework, even for beginners*. https://www.semanticscholar.org/paper/192358d116198f8f69f5c3fc6da7d9b6aa01f21c

Mrdoob. (2020). *three.js*. Github. https://github.com/mrdoob/three.js/

Müller, C., Gärtner, F., & Dik, D. (2014). Portable 3D-browser-applications using cross-compiled programming languages. *Proceedings - 2014 International Conference on Computational Science and Computational Intelligence, CSCI 2014*, *2*, 229–232. https://doi.org/10.1109/CSCI.2014.125

Naciones Unidas. (2018). *Infraestructura - Desarrollo Sostenible*. Naciones Unidas. https://www.un.org/sustainabledevelopment/es/infrastructure/

Netlify. (2021). *Netlify*. Netlify. https://www.netlify.com

Oliphant, T. E. (2010). Guide to NumPy. *Methods*, *1*, 378. http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Guide+to+NumPy#0

Pandas. (2020). *About pandas*. Pandas.Pydata.Org. https://pandas.pydata.org/about/index.html

Peluffo-Ordónez, C. K. B.-V. C. M. O.-C. D. F. P.-U. E. J. R.-F. J. A. S.-C. M. O.-B. P.

R.-M. L. S. V.-E. D. H. (2018). *Angle-Based Model for Interactive Dimensinality Reduction and Data Visualization*. *1*(October), 201–209. https://doi.org/10.1007/978-3-030-01132-1_17

Peluffo-Ordóñez, D. H., Alvarado-Pérez, J. C., Lee, J. A., & Verleysen, M. (2015a). Geometrical homotopy for data visualization. *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2015 - Proceedings*, *April*, 525–530.

Peluffo-Ordóñez, D. H., Alvarado-Pérez, J. C., Lee, J. A., & Verleysen, M. (2015b). Geometrical homotopy for data visualization. *23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, ESANN 2015 - Proceedings*, 525–530.

Peña-Unigarro, D. F., Rosero-Montalvo, P., Revelo-Fuelagán, E. J., Castro-Silva, J. A., Alvarado-Pérez, J. C., Therón, R., Ortega-Bustamante, C. M., & Peluffo-Ordóñez, D. H. (2017). Interactive data visualization using dimensionality reduction and dissimilarity-based representations. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *10585 LNCS*, 461–469. https://doi.org/10.1007/978-3-319-68935-7_50

Pena-Unigarro, D. F., Salazar-Castro, J. A., Peluffo-Ordonez, D. H., Rosero-Montalvo, P. D., Ona-Rocha, O. R., Isaza, A. A., Alvarado-Perez, J. C., & Theron, R. (2016). Interactive visualization methodology of high-dimensional data with a color-based model for dimensionality reduction. *2016 21st Symposium on Signal Processing, Images and Artificial Vision, STSIVA 2016*, 1–7. https://doi.org/10.1109/STSIVA.2016.7743318

plotly. (2020a). *Plotly 3D Subplots*. Github. https://plotly.com/javascript/3d-subplots/

plotly. (2020b). *Plotly Fundamentals*. Github. https://plotly.com/javascript/plotly-fundamentals/

plotly. (2020c). *Plotly JavaScript Graphic Library*. Github. https://plotly.com/javascript/

Python. (2021). *Python*. Python.Org. https://www.python.org/doc/essays/blurb/

Rosero-Rosero, P. ., Peña-Unigarro, D. F., Peluffo, D. H., Castro-Silva, J. A., A. Umaquinga, & Rosero-Rosero, E. A. (2017). Data Visualization Using Interactive Dimensionality Reduction and Improved Color-Based Interaction Model. *Lecture Notes in Computer Science*, *2*, 289–298. https://doi.org/10.1007/978-3-319-59773-

7

Sacha, D., Zhang, L., Sedlmair, M., Lee, J. A., Peltonen, J., Weiskopf, D., North, S. C., & Keim, D. A. (2017). Visual Interaction with Dimensionality Reduction: A Structured Literature Analysis. *IEEE Transactions on Visualization and Computer Graphics*, *23*(1), 241–250. https://doi.org/10.1109/TVCG.2016.2598495

Salazar-Castro, J. A., Rosas-Narvaez, Y. C., Pantoja, A. D., Alvarado-Perez, J. C., & Peluffo-Ordonez, D. H. (2015). Interactive interface for efficient data visualization via a geometric approach. *2015 20th Symposium on Signal Processing, Images and Computer Vision, STSIVA 2015 - Conference Proceedings*. https://doi.org/10.1109/STSIVA.2015.7330397

Salazar-Castro, Jose A., Peña-Unigarro, D., Peluffo-Ordóñez, D. H., Rosero-Montalvo, P. D., Dominguez-Limaico, H. M., Alvarado-Perez, J. C., & Theron, R. (2017a). Dimensionality reduction for interactive data visualization via a Geo-Desic approach. *2016 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2016 - Proceedings*, 1–6. https://doi.org/10.1109/LA-CCI.2016.7885740

Salazar-Castro, Jose A., Peña-Unigarro, D., Peluffo-Ordóñez, D. H., Rosero-Montalvo, P. D., Dominguez-Limaico, H. M., Alvarado-Perez, J. C., & Theron, R. (2017b). Dimensionality reduction for interactive data visualization via a Geo-Desic approach. *2016 IEEE Latin American Conference on Computational Intelligence, LA-CCI 2016 - Proceedings*. https://doi.org/10.1109/LA-CCI.2016.7885740

Samet, H. (2005). *Foundations of Multidimensional and Metric Data Structures (The Morgan Kaufmann Series in Computer Graphics and Geometric Modeling)*.

Sauro, J. (2018). *5 Ways to Interpret a SUS Score*. Measuringu.Com. https://measuringu.com/interpret-sus-score/

StackOverflow. (2020). *StackOverflow 2020 Developer Survey*. Insights.Stackoverflow.Com. https://insights.stackoverflow.com/survey/2020#technology-programming-scripting-and-markup-languages

Statcounter. (2021). *Browser Market Share Worldwide*. https://gs.statcounter.com/browser-market-share#monthly-202005-202105

Sterling, A. (2019). NodeJS and Angular Tools for JSON-LD. *Proceedings - 13th IEEE International Conference on Semantic Computing, ICSC 2019*, 392–395. https://doi.org/10.1109/ICOSC.2019.8665625

Systeminformation. (2021). *Systeminformation*. Systeminformation. https://systeminformation.io/gettingstarted.html

Therón;, P. R.-M. P. D. J. A. S.-C. D. F. P.-U. A. J. A. I. J. . A.-P. R. (2017). *Interactive Data Visualization Using Dimensionality Reduction and Similarity-Based Representations*. *10125*, 10–18. https://doi.org/10.1007/978-3-319-52277-7

Three.js. (2020a). *three.js example*. Three.Js. https://threejs.org/examples/#webgl_animation_cloth

Three.js. (2020b). *WebGL Performance*. Three.Js. https://threejs.org/examples/?q=per#webgl_performance

Three.js. (2021). *Shaders/Ocean*. https://threejs.org/examples/?q=ocea#webgl_shaders_ocean

Wang, S. (2021). *node-os-utils*. NPM. https://www.npmjs.com/package/node-os-utils

Webpack. (2021). *Webpack*. Webpack. https://webpack.js.org/concepts/

World Wide Web Consortium. (2020). *HTML & CSS*. W3C. https://www.w3.org/standards/webdesign/htmlcss.html

Wulandari, D. A. P., & Aristana, M. D. W. (2021). Analysis Evaluation Management Information System Audit Internal Quality. *Journal of Electrical, Electronics and Informatics*, *5*(1), 5. https://doi.org/10.24843/jeei.2021.v05.i01.p02

ANEXOS

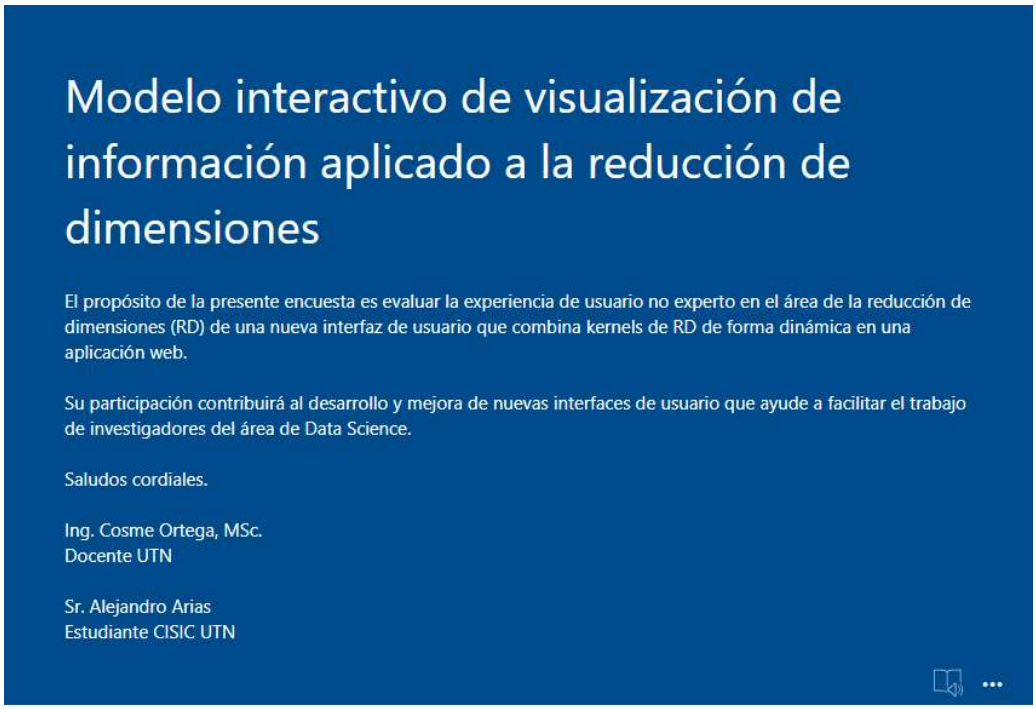
Anexo A: Datos en crudo analizados en la comparativa de rendimiento de librerías de renderizado 3D

Se generaron 45 archivos *logs* en las pruebas de rendimiento, dichos *logs* se encuentras disponibles en el siguiente *gist*:

https://gist.github.com/Disble/48347338b60e087324238ebb06be6278

Anexo B: Encuesta de usabilidad SUS

A continuación, se presenta la encuesta SUS realizada en la validación de resultados del Modelo Interactivo.



* Required

Modelo interactivo de visualización de información aplicado a la reducción de dimensiones

1. Sistema de Escala de Usabilidad *

	Totalmente en desacuerdo	En desacuerdo	Ni de acuerdo, ni en desacuerdo	De acuedo	Totalmente de acuerdo
1. Creo que me gustaría utilizar frecuentemente este sitio web.	○	○	○	○	○
2. Encontré el sitio web sencillo.	○	○	○	○	○
3. Pienso que el sitio web es fácil de usar.	○	○	○	○	○
4. Pienso que podré utilizar este sitio web sin el apoyo de personal técnico.	○	○	○	○	○
5. Encontré que varias de las funciones en el sitio en el sitio web estaban bien integradas.	○	○	○	○	○
6. Pensé que había demasiada consistencia en el sitio web.	○	○	○	○	○
7. Me imagino que la mayoría de las personas podrían aprender a usar este sitio web muy rápido.	○	○	○	○	○
8. Encontré el sitio web muy intuitivo.	○	○	○	○	○
9. Me sentí muy confiado (seguro) al utilizar el sitio web.	○	○	○	○	○
10. Pude utilizar el sitio web sin tener que aprender nada nuevo.	○	○	○	○	○