UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

Desarrollo de una experiencia de realidad aumentada para un proceso de tanques atmosféricos interconectados

T E S I S

Que para obtener el título de

Ingeniero Eléctrico Electrónico

P R E S E N T A

Bruno Alejandro Rodríguez Jiménez

DIRECTOR DE TESIS

Dr. Hoover Mujica Ortega

Ciudad Universitaria, Cd. Mx., 2026

PROTESTA UNIVERSITARIA DE INTEGRIDAD Y HONESTIDAD ACADÉMICA Y PROFESIONAL
(Titulación con trabajo escrito)

De conformidad con lo dispuesto en los artículos 87, fracción V, del Estatuto General, 68, primer párrafo, del Reglamento General de Estudios Universitarios y 26, fracción I, y 35 del Reglamento General de Exámenes, me comprometo en todo tiempo a honrar a la institución y a cumplir con los principios establecidos en el Código de Ética de la Universidad Nacional Autónoma de México, especialmente con los de integridad y honestidad académica.

De acuerdo con lo anterior, manifiesto que el trabajo escrito titulado DESARROLLO DE UNA EXPERIENCIA DE REALIDAD AUMENTADA PARA UN PROCESO DE TANQUES ATMOSFERICOS INTERCONECTADOS que presenté para obtener el titulo de INGENIERO ELÉCTRICO ELECTRÓNICO es original, de mi autoría y lo realicé con el rigor metodológico exigido por mi Entidad Académica, citando las fuentes de ideas, textos, imágenes, gráficos u otro tipo de obras empleadas para su desarrollo.

En consecuencia, acepto que la falta de cumplimiento de las disposiciones reglamentarias y normativas de la Universidad, en particular las ya referidas en el Código de Ética, llevará a la nulidad de los actos de carácter académico administrativo del proceso de titulación.

BRUNO ALEJANDRO RODRIGUEZ JIMENEZ
Número de cuenta: 419049386

Jurado asignado

Presidente:	M.I. Ulises Martín Peñuelas Rivas
Secretaria:	Dra. María del Pilar Corona Lira
Vocal:	Dr. Hoover Mujica Ortega
1^{er} suplente:	M.I. Iván de Jesus Osio Chávez
2^{do} suplente:	Ing. David Quintanar Villalba

Ciudad Universitaria, Departamento de Control y Robótica, Laboratorio de Automatización.

Ciudad de México.

Director de tesis

Dr. Hoover Mujica Ortega

Dedicatoria

Este trabajo está dedicado a todos aquellos que formaron parte del viaje de mi formación académica, y que, con su presencia, palabras de aliento, enseñanzas y motivación, me han impulsado a estudiar y concluir la carrera que soñé desde la infancia.

Agradecimientos

Agradezco profundamente a mis padres, quienes de manera incondicional me brindaron su apoyo emocional y material para alcanzar esta meta. A mi familia, por confiar en mis capacidades, por sus palabras de aliento en los momentos difíciles y por compartir conmigo los logros alcanzados.

Un agradecimiento especial para mi compañera de aventuras, cuyo apoyo, afecto y amistad han trascendido el tiempo de esta carrera y fueron esenciales para llegar aquí.

Agradezco también a mis compañeros de la carrera y del servicio social en el Laboratorio de Automatización, con quienes pude apoyarme para elevar la calidad de mi trabajo y mis conocimientos.

Finalmente, extiendo mi sincero agradecimiento al Dr. Hoover Mujica por su invaluable guía académica, su compromiso con mi formación profesional y su constante disposición para orientar este trabajo, así como a la Facultad de Ingeniería por brindarme una educación de alta calidad que ha fortalecido mi vocación y pasión por la ingeniería.

Resumen

En el marco de esta tesis, se desarrolló e implementó una experiencia de realidad aumentada (AR, por sus siglas en inglés) aplicada a una instalación industrial de manejo de líquidos, consistente en un sistema de tuberías instrumentado que interconecta dos tanques de almacenamiento no presurizados. La aplicación fue desarrollada en el *software* industrial Vuforia Studio, el cual, en conjunto con la aplicación multiplataforma Vuforia View, permitió crear una herramienta interactiva sobre el sistema descrito, caracterizada por ser descriptiva, escalable, accesible en distintos sistemas operativos y de fácil distribución.

Este trabajo plantea que la AR puede ser una herramienta de asistencia para ingenieros y operadores en entornos industriales, con potencial para mejorar tareas de mantenimiento, ensamble, operación y capacitación en instalaciones de este tipo.

Motivado por lo anterior, el objetivo de esta investigación fue desarrollar, implementar y validar una experiencia de AR aplicada al sistema descrito. Además, se realizó una evaluación técnica de la aplicación para asegurar su rendimiento, usabilidad y calidad visual. para lo cual se consideraron métricas técnicas como tiempos de respuesta, consumo de recursos computacionales (memoria de acceso aleatorio (RAM, por sus siglas en inglés), unidad de procesamiento gráfico (GPU, por sus siglas en inglés) y unidad central de procesamiento (CPU, por sus siglas en inglés)) y aspectos de compatibilidad, accesibilidad y calidad visual.

El documento presenta y analiza los conceptos teóricos necesarios para comprender el desarrollo de experiencias en AR. Asimismo, se detallan las etapas del proceso de diseño, junto con las herramientas utilizadas para crear animaciones, secuencias, interacciones, controles y elementos visuales.

Como resultado, se desarrolló una experiencia de AR que utiliza la pantalla de un teléfono celular o tableta para superponer modelos tridimensionales de los componentes de la instalación, así como información técnica y animaciones de funcionamiento, sobre superficies y objetos del entorno real.

Índice general

Índice de figuras

Índice de tablas

Acrónimos

AR Realidad aumentada. IX, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21, 22, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 38, 45, 49, 50, 53, 61, 63, 71, 74, 78, 79, 81, 88, 91, 92
RAM Memoria de acceso aleatorio. IX, 14, 23, 82, 83, 85, 86, 87, 88
GPU Unidad de procesamiento gráfico. IX, 14, 23, 82, 83, 85, 88, 89
CPU Unidad central de procesamiento. IX, 14, 50, 82, 83, 85, 86, 87, 88
PLC Controlador lógico programable. 3, 46
VR Realidad virtual. 7, 17
IoT Internet de las cosas. 10, 21, 25, 30, 31, 33, 35
SCADA Sistemas de supervisión, control y adquisición de datos. 10, 46
CAD Diseño asistido por computadora. 10, 15, 27, 28, 29, 30, 31, 33, 35, 46, 49, 50, 51, 52, 53, 62, 63, 69, 71, 78, 81, 82, 89
SDK Kit de desarrollo de software. 11, 28, 29, 30, 82
FPS Fotogramas por segundo. 14, 82, 83, 86, 87, 88
CSS Hojas de estilo en cascada. 15, 35, 40, 41, 43, 53, 54, 55, 56, 58, 59, 69
GPS Sistema de posicionamiento global. 17, 18, 20
UI Interfaz de usuario. 20, 21, 33, 52, 55, 89
IT Tecnologías de la información. 21, 22, 25
PSS Sistema de producto-servicio. 21, 26
IA Inteligencia artificial. 21
HTML Lenguaje de marcado de hipertexto. 34, 35, 40, 56
PLM Administración del ciclo de vida del producto. 35, 92
HMI Interfaz humano-máquina. 46
PC Computadora personal. 61, 81, 82, 83, 85, 88

Capítulo 1

Introducción

Actualmente, la sociedad experimenta una revolución digital que transforma continuamente la forma en que las personas interactúan con el entorno y realizan diversas actividades. En este sentido, las tecnologías disruptivas han impactado varios aspectos de la vida, desde las relaciones personales hasta la educación y la industria. Es así que la AR se presenta como una herramienta innovadora con el potencial de enriquecer y transformar la percepción e interacción con el mundo y, por lo tanto, los métodos y herramientas de trabajo.

La presente investigación explora cómo la AR, en el contexto de la Industria 4.0, puede impactar positivamente en el área operativa de sistemas industriales y productivos. Para comprender este potencial, se examinarán las características y usos de esta innovación; posteriormente, se delimitará el campo de aplicación previsto: la operación y automatización de instalaciones industriales. Esto permitirá identificar claramente áreas de oportunidad y carencias en las que dicha tecnología podría aportar valor.

Introducción a la realidad aumentada

La AR es una herramienta que enriquece la experiencia sensorial de las personas, al combinar ambientes virtuales con el mundo físico mediante dispositivos tecnológicos como teléfonos móviles, tabletas, monitores, lentes de AR u otros dispositivos capaces de superponer información virtual sobre el entorno real. Una definición formal de esta tecnología es: "un medio en el cual la información digital se superpone en el mundo físico, en correspondencia espacial y temporal con el espacio físico y que es interactivo en tiempo real" [Craig, 2013, p.15]. Con base en esta definición, se pueden enumerar algunas características de la AR:

- Potencia los sentidos humanos, principalmente con estímulos visuales y auditivos.
- Es inmersiva, ya que la superposición en tiempo real de imágenes, marcadores o información generada virtualmente crea un entorno enriquecido que el usuario percibe como parte de su realidad tangible [Telefónica, 2011].
- Es interactiva, pues las acciones del usuario influyen directamente en la representación de la realidad mostrada en el dispositivo.

Un ejemplo de AR que reúne todas las características mencionadas anteriormente es el uso de visores y tabletas para mostrar información en entornos industriales. En este tipo de aplicaciones, el

dispositivo superpone la imagen tridimensional de una pieza concreta de una máquina sobre el entorno real captado por la cámara, aumentando con datos virtuales la representación de la realidad mostrada en la pantalla, tal como se muestra en la Figura 1.1. Esto proporciona estímulos visuales al usuario, quien, al observar la pieza, realiza acciones con el dispositivo, como tocar la pantalla, seleccionar menús emergentes o maniobrar el equipo, entre otras. Dichas interacciones activan animaciones, sonidos y efectos visuales dentro de la aplicación, lo que permite al usuario experimentar un entorno enriquecido, en el cual puede percibir e interactuar con elementos virtuales sin dejar de observar el mundo que lo rodea.

Figura 1.1 Ejemplo de AR en un entorno industrial.[1]

Gracias a estas características, la AR tiene una amplia gama de aplicaciones, tanto en sectores cotidianos como en empresariales e industriales. Como se menciona en [Telefónica, 2011], algunos de los campos más importantes son:

- **Ocio:** Videojuegos, películas, libros y experiencias inmersivas en museos mediante AR.
- **Ventas:** Visualización de artículos en su ubicación final (muebles, decoración) o simulación de uso por parte del cliente (ropa, calzado deportivo, celulares).
- **Industrial:** Manufactura, mantenimiento y capacitación de empleados en habilidades técnicas.

Instalaciones industriales y automatización

Las instalaciones industriales específicas forman parte de las instalaciones físicas y equipos de una planta, que permiten la ejecución de procesos de producción o servicios complementarios. Estas instalaciones incluyen sistemas físicos, eléctricos, mecánicos y de proceso, cuya instalación, operación y mantenimiento constituyen funciones fundamentales de la operación de una planta [Mobley, 2001]. Algunos ejemplos son: sistemas de manejo de fluidos, refrigeración, generación de vapor, almacenamiento, entre otros.

Por su parte, la automatización de un proceso industrial busca que un sistema opere de forma autónoma, respondiendo a las variaciones del proceso mediante sistemas de control. El objetivo de la automatización es asegurar que la instalación cumpla su función de manera más segura, eficiente y confiable que mediante operación manual [Sanchis Llopis*, et al.*, 2010], así como reducir los costes

[1] Tomado de [BMW Group, 2020].

de fabricación, mantener una calidad constante en la producción y liberar a los humanos de tareas tediosas, peligrosas o insalubres [Escaño González*, et al.*, 2019].

La creciente competitividad empresarial, impulsada por la Industria 4.0, ha demandado sistemas de producción más flexibles, capaces de ajustar la estrategia de producción según las necesidades del proceso, lo que expande los objetivos y retos de la automatización. Según [Moreno, 2001], estos retos se pueden abordar mediante la gestión efectiva de los tres niveles de decisión en un proceso productivo: estratégico, táctico y operacional. Este último nivel se ocupa del control cotidiano de las operaciones de fabricación [Moreno, 2001], es decir, tareas de mantenimiento, reparación, operación y supervisión de instalaciones industriales específicas a nivel de planta.

En este mismo contexto, autores como [Escaño González*, et al.*, 2019], destacan la importancia de que los profesionales encargados de proyectos de automatización, como los integradores de sistemas automáticos o ingenieros de control, comprendan, al menos de forma general, el funcionamiento de las instalaciones que buscan automatizar. Esto también se constata en la norma IEC 61508, titulada "Seguridad funcional de sistemas eléctricos/electrónicos/programables electrónicos relacionados con la seguridad", que se aplica a estos sistemas desde su diseño hasta la operación y el mantenimiento. Particularmente, en la sección 6.2.13 se menciona que los profesionales deben contar con el conocimiento técnico, experiencia y cualificaciones para desarrollar las tareas de esta área [Comisión Electrotécnica Internacional, 2010].

Lo anterior subraya la necesidad de herramientas que conecten el conocimiento teórico de la automatización, como la programación en un controlador lógico programable (PLC, por sus siglas en inglés), las estrategias de control y el desarrollo de pantallas de supervisión, con el entendimiento de las características y el funcionamiento de las instalaciones industriales. En este sentido, la AR puede ser un complemento valioso, con el potencial de motivar, facilitar y optimizar la integración de estos conocimientos gracias a su naturaleza inmersiva.

Una vez presentado el contexto de esta investigación, la siguiente sección aborda las razones para considerar que la integración de AR en procesos productivos puede ayudar a superar los retos y objetivos actuales de la automatización y auxiliar en las tareas del área operativa de instalaciones industriales.

1.1. Motivación

Las tareas de operación, mantenimiento y ensamblaje en el piso de planta representan un campo estratégico clave donde la AR puede aportar un valor significativo para lograr los objetivos de la automatización. Sin embargo, en la operación diaria de las instalaciones industriales surgen desafíos que pueden dificultar la consecución de estas metas. Muchos de estos problemas están relacionados con la complejidad de las tareas que los operadores deben realizar, quienes a menudo necesitan consultar manuales escritos para llevar a cabo los procedimientos correctamente. El constante cambio de atención provocado por alternar entre realizar tareas y consultar el manual aumenta el nivel de concentración necesario y la carga cognitiva, lo que incrementa el riesgo de cometer errores [Hou*, et al.*, 2013].

Según el mismo autor, el área de ensamblaje es particularmente vulnerable a estos problemas, pues allí el operador debe realizar tareas físicas (como observar, sujetar e instalar) y procesos cognitivos (como comprender, traducir, recuperar y aplicar información); como resultado, la combinación y sincronización de estas actividades consume mucho tiempo y exige frecuentes cambios de atención. Sobre esta misma línea, [Towne, 1985] señala que las actividades relacionadas con la información representan el 50 % del tiempo invertido en una tarea; por su parte [Veinott y Kanki, 1995] revelan que los ensambladores dedican el 45 % de su tiempo a buscar y leer información procedimental, y que el 60 % de los errores cometidos durante el ensamblaje se deben a malentendidos del manual.

Estas cifras sugieren que, tanto el tiempo dedicado a las tareas como la frecuencia de los errores, son áreas que pueden optimizarse incorporando herramientas que faciliten la consulta de instrucciones y reduzcan la carga cognitiva, mejorando así la eficiencia y seguridad del proceso de ensamblaje.

En paralelo, las áreas de operación y mantenimiento de instalaciones industriales enfrentan desafíos similares, ya que también dependen del uso de manuales estáticos impresos en papel y requieren una coordinación entre acciones físicas y trabajo cognitivo. En adición a esto, cada una de estas áreas presenta retos específicos adicionales; en el trabajo de [Garza*, et al.*, 2013] se dice que el mantenimiento es crucial en los procesos productivos, ya que las fallas en la maquinaria pueden causar pérdidas significativas para las empresas; por lo tanto, es esencial realizar estas tareas de manera constante y eficiente. Sin embargo, algunas dificultades para lograrlo son la necesidad de detener completamente las líneas de producción durante el mantenimiento, lo que hace crítico minimizar el tiempo de inactividad. Además, cuando se requiere un especialista, la programación de visitas puede ser difícil debido a limitaciones de tiempo, y la asesoría telefónica a menudo no es suficiente para transmitir el conocimiento de manera efectiva y rápida.

Aunado a lo anterior, otro sector fundamental para asegurar la eficiencia y seguridad en todo proceso productivo es el entrenamiento de habilidades de operadores, ingenieros de control y personal de mantenimiento. Sin embargo, al analizar instalaciones industriales reales, se identifican carencias en los métodos tradicionales de capacitación que dificultan la transmisión efectiva del conocimiento. Según [Larson, 2022] los métodos convencionales, como conferencias y materiales escritos, suelen ser menos efectivos para lograr una comprensión y retención adecuadas de la información por parte de los participantes; estos enfoques tradicionales a menudo fallan en involucrar activamente a los involucrados, lo cual es crucial para un aprendizaje más eficaz.

Asimismo, derivado de observaciones personales realizadas en el contexto de instalaciones industriales complejas, se ha notado que el acceso y estudio de los equipos a menudo se ve obstaculizado por factores como la seguridad, la falta de visibilidad y la imposibilidad de detener ciertas máquinas.

Todas estas problemáticas y oportunidades de mejora reflejan la necesidad de herramientas que vinculen el conocimiento teórico con la experiencia práctica y que optimicen las tareas del sector

operacional de instalaciones industriales. Para enfrentar estos desafíos, las tendencias actuales en los sistemas productivos se enfocan en la integración de nuevas tecnologías; el objetivo no es solo superar los problemas mencionados, sino también construir sistemas productivos más resilientes, ágiles y sostenibles. Esta tendencia ha sido descrita como "transformación digital" por empresas de automatización como [Rockwell Automation, 2023]. La AR (ubicada del lado derecho de la Figura 1.2), al ser una tecnología emergente, se alinea perfectamente con dicha tendencia.

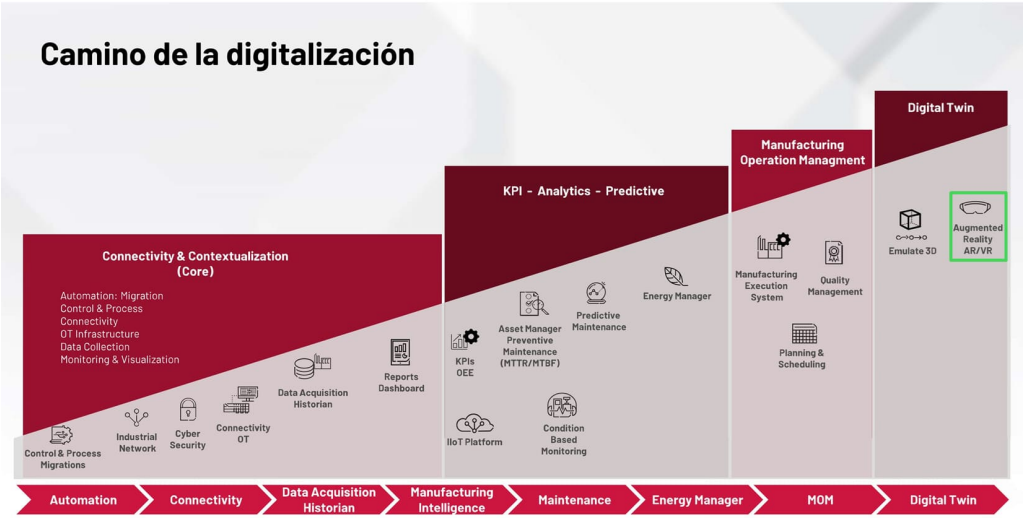


Figura 1.2 Diagrama del camino de la digitalización. [2]

El valor potencial de esta tecnología en los sectores productivos está respaldado por varias razones: algunas empresas líderes en el área la ven como una herramienta clave en los nuevos paradigmas de fabricación; las tendencias de investigación en mantenimiento, operación y capacitación muestran una creciente integración de esta tecnología en los procesos industriales y su madurez actual promete una incorporación eficiente, segura y rápida en dichos procesos. Los siguientes párrafos profundizarán en estos temas, los cuales han motivado la exploración del potencial de la AR como a las soluciones para los problemas descritos anteriormente.

Como menciona [Rockwell Automation, 2023] en la presentación "Innovación, productividad y sostenibilidad para la Fabricación del Futuro", las tecnologías emergentes son fundamentales para acelerar la transformación digital. Este potencial se representa en el diagrama mostrado en la Figura 1.2; allí se pueden observar algunas de las tecnologías y estrategias que esta empresa considera clave. Señalada en color verde, se encuentra la AR; su aparición en este gráfico la coloca como una de las herramientas que pueden ayudar a superar los problemas planteados anteriormente y alcanzar los nuevos estándares de fabricación establecidos por la Industria 4.0 y la fabricación inteligente.

Además, la madurez tecnológica alcanzada por la AR está respaldada por el esquema del *hype cycle* de la empresa de consultoría tecnológica Gartner; esta representación gráfica sirve para describir la evolución y maduración de tecnologías emergentes con el potencial para convertirse en innovaciones disruptivas [Shrivastava, 2021]. Gartner realiza la valoración de las tecnologías en dos dimensiones; la primera es el nivel de expectativas que generan en potenciales clientes y la sociedad, representadas en el eje de las ordenadas. La segunda se relaciona con el tiempo, ubicado en el eje de las abscisas de

[2] Tomado de [Rockwell Automation, 2023].

la gráfica, y abarca el periodo desde el nacimiento de la tecnología hasta que esta se consolida. Una explicación de las 5 etapas que componen el ciclo se encuentra en la Figura 1.3:

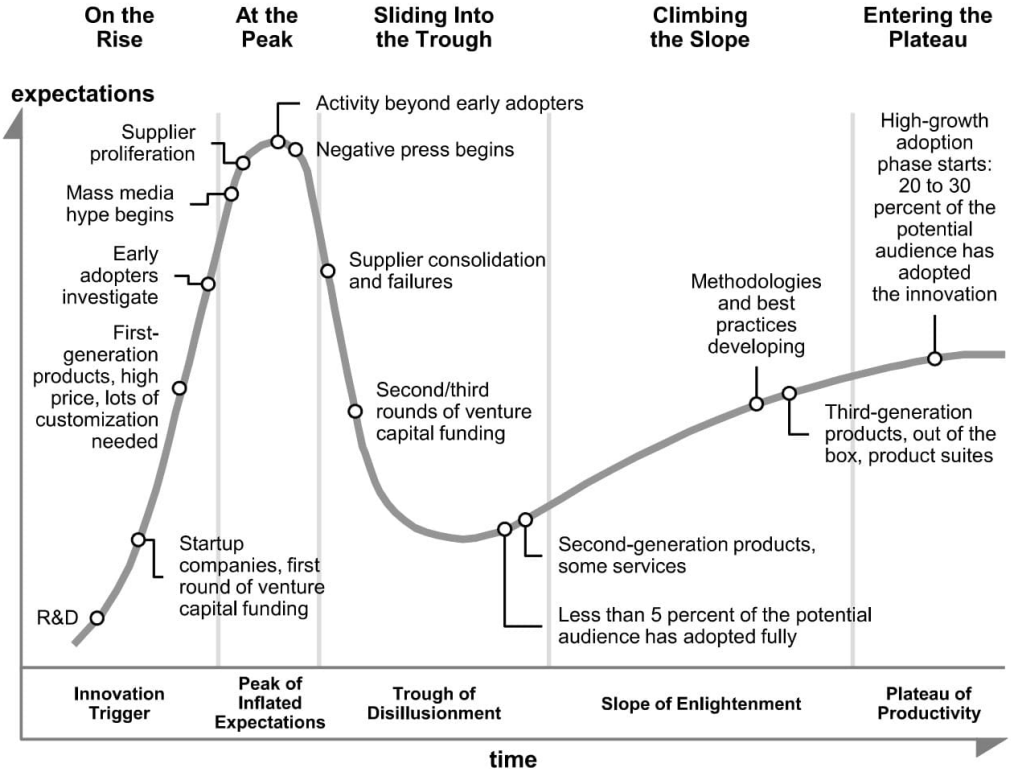


Figura 1.3 Explicación de las etapas del *hype cycle*. [3]

El *hype cycle* le ha dado seguimiento a la evolución y maduración de la AR; en la Figura 1.4 se observa su recorrido, desde la primera aparición en el diagrama en el 2005 hasta su salida en el año 2018. Como se puede ver, entre los años 2010-2011 esta tecnología vivió el pico de expectativas; en esa época ya se teorizaba sobre la explotación industrial de esta innovación, tal como se hace en el artículo "Realidad aumentada: ¿revolución o promesa vana?" de 2011 [BBCMundo, 2011].

Los desarrollos y avances en esta área continuaron de la mano de empresas como PTC, Qualcomm, Google, entre otras, las cuales lograron crear motores de desarrollo de experiencias, tales como Wikitude, Vuforia, ARKit, ARCore, XCode 7, entre otras, [Dan Miller, 2024], [PTC, 2024c], [Peterson, 2021], cimentando así las bases para que esta tecnología entrara a la rampa de consolidación.

Posteriormente, la AR no fue incluida en la edición del *hype cycle* de Gartner de 2019; según palabras de la propia empresa, el motivo fue que dejó de ser una innovación emergente; por consiguiente, evolucionó a un estado maduro, convirtiéndose en una tecnología confiable y útil para la industria, en la cual se puede invertir de manera segura para mejorar e innovar en los procesos [Herdina, 2020].

Finalmente, otro motivo para la exploración de las oportunidades al usar AR industrial es el interés creciente de los sectores productivos para adoptar esta tecnología, lo que se puede ver reflejado en las tendencias de las investigaciones relacionadas con los sistemas de manufactura industriales. En el estudio "*The Role of Maintenance Operator in Industrial Manufacturing Systems: Research Topics*

[3]Tomado de [Shrivastava, 2021].
[4]Obtenido de: [Herdina, 2020].

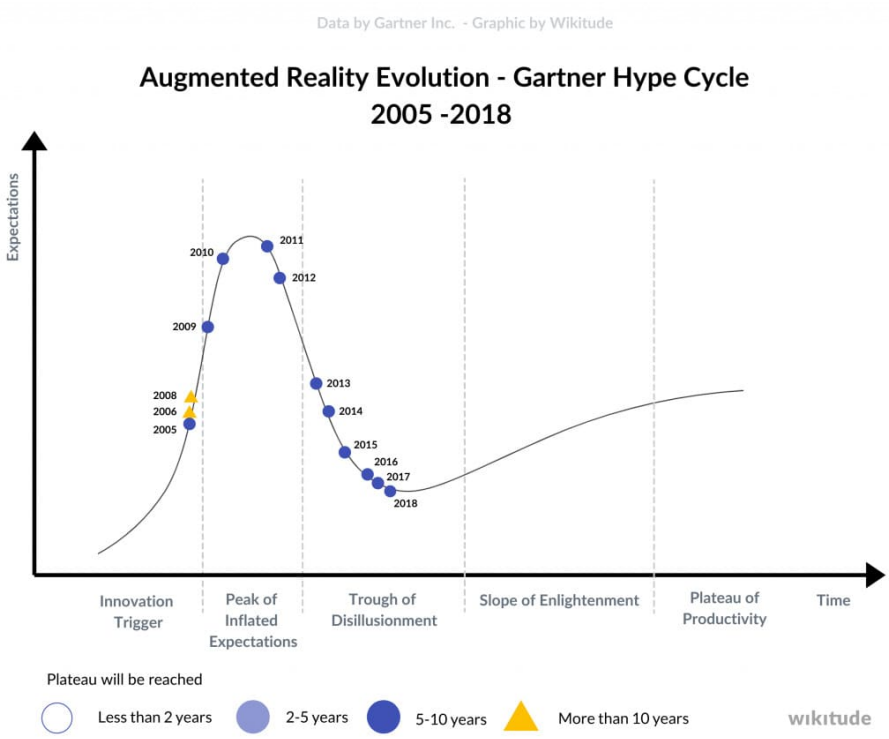


Figura 1.4 Diagrama del *hype cycle* de la AR. [4]

and Trends" [Tortora, *et al.*, 2021], se analizaron las palabras clave y temas principales de 79 artículos académicos relacionados con el mantenimiento industrial, previamente filtrados para ser relevantes.

Como se puede ver en la Figura 1.5, los resultados revelaron que el 33 % de los estudios estaban centrados en la integración de realidad virtual (VR, por sus siglas en inglés) y AR en procedimientos de mantenimiento, de los cuales 20 trabajos se enfocaron en la segunda tecnología.

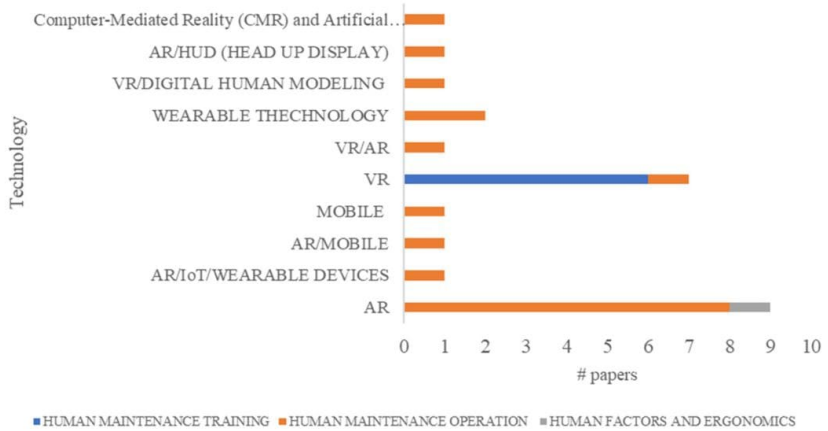


Figura 1.5 Aparición de la AR en estudios sobre mantenimiento industrial. [5]

Los autores concluyeron que el uso de AR en dispositivos móviles o portátiles mejora significativamente las tareas de mantenimiento, aumentando la productividad y reduciendo tiempo y errores.

[5] Obtenido de: [Tortora, *et al.*, 2021].

Además, observaron una tendencia creciente en la investigación hacia la reducción de costos de mantenimiento, la mejora del rendimiento del operador y la facilitación del entrenamiento mediante instrucciones más accesibles y fáciles de entender para operadores sin experiencia [Tortora*, et al.*, 2021].

Basado en los motivos expuestos en esta sección, y con el objetivo de aprovechar estas bondades de la tecnología, se propone el desarrollo de una experiencia de AR para una instalación industrial de manejo de líquidos, la cual se compone de dos tanques atmosféricos interconectados mediante tuberías, controlados por válvulas analógicas y digitales accionadas neumáticamente, instrumentada con medidores de presión, flujo y nivel, y operado a través de bombas eléctricas trifásicas [Santamaría, 2024].

El sistema constantemente es operado por ingenieros de automatización y, al ser una instalación industrial, enfrenta los desafíos previamente descritos. La integración de una experiencia de AR en este entorno busca optimizar la realización de las tareas operativas, proporcionar a los usuarios una herramienta para comprender detalladamente el sistema y alinear a los colaboradores con las tendencias tecnológicas del sector.

1.2. Antecedentes

Como se mencionó en los subcapítulos anteriores, existen numerosos antecedentes sobre el uso de la AR en el ámbito industrial. En esta sección, se analizarán aplicaciones centradas en el área operativa de sistemas productivos; en primer lugar, se presentan investigaciones publicadas en revistas especializadas y, posteriormente, implementaciones reales en la industria. Finalmente, se hará un breve recuento de los entornos para desarrollar experiencias de AR y se evaluará cuál de ellos es más adecuado para los objetivos de esta investigación.

1.2.1. Estudios y experimentos sobre AR industrial

Según [Morales Méndez y del Cerro Velázquez, 2024] la mayoría de estudios acerca de aplicaciones industriales de la AR se concentran en las áreas de ensamblaje, mantenimiento, monitoreo de la calidad, entre otras. Derivado de estos estudios, se han podido identificar los siguientes beneficios potenciales de esta tecnología:

- **Mejora el entendimiento de sistemas industriales:** La AR puede presentar información en un formato que integra tanto el aspecto espacial como el temporal. Este enfoque resulta más efectivo para el aprendizaje en comparación con los formatos no integrados [Mayer y Fiorella, 2014], lo que afecta positivamente el desempeño y la carga cognitiva [Thees*, et al.*, 2020]. Esta tecnología es particularmente útil para comprender el funcionamiento de maquinaria o sistemas de mecanismos; además, puede aumentar la motivación para aprender [Tortora*, et al.*, 2021]. De la misma forma, en el trabajo de [Longo*, et al.*, 2017] se menciona que las pendientes típicas de aprendizaje para operaciones de mecanizado están en el rango del 90 al 95 % con AR. Por su parte, la tasa promedio de aprendizaje lograda mediante entrenamiento tradicional es de 91.85 %, demostrando que la retención de información mejora en el primer caso.
- **Aumenta la seguridad y eficiencia:** En el estudio llevado a cabo por [Uva*, et al.*, 2018] se encontró que, a través de tecnologías de AR, la efectividad para llevar a cabo tareas con instrucciones técnicas mejoró en términos de tiempo (una reducción general del 20.3 %) y tasa de errores cometidos (83.3 % menos errores). De la misma forma [Papakostas*, et al.*, 2022] afirman que los simuladores de AR para el entrenamiento de soldadores no tienen riesgos físicos, no implican emisiones de gases nocivos y conllevan un ahorro significativo en gastos y tiempo invertido en la capacitación, demostrando cómo esta tecnología hace más seguros y eficientes los procesos industriales. Según [Longo*, et al.*, 2017] estos resultados son alcanzables proveyendo a los operadores con información que usualmente no está disponible en su lugar de trabajo (índices de productividad, advertencias sobre peligros, instrucciones para el operador), así como advertencias sobre las consecuencias de operar inadecuadamente un equipo. En este estudio se reportó una tasa de reducción en el tiempo de ejecución de tareas manuales del 10.18 % en comparación con un 8.15 % alcanzado mediante capacitación tradicional cuando la producción se duplica.
- **Permite la comunicación en tiempo real:** La comunicación en tiempo real entre el sistema de planificación y monitoreo, junto con el uso de tecnología de AR, mejora la eficiencia de un operador de mantenimiento y, por ende, aumenta el rendimiento total del sistema [Tortora*, et al.*, 2021]. Según la implementación presentada por [Mourtzis*, et al.*, 2017], gracias a esta combinación, la

conciencia sobre el estado de la maquinaria industrial puede aumentar un 30 % y la respuesta ante cambios imprevistos en el funcionamiento puede reducirse en más del 25 %. Además, la posibilidad de acceder a la visión de un operador en piso de planta permite nuevas aplicaciones, como la supervisión de calidad en tiempo real [De Pace*, et al.*, 2018].

- **Permite el monitoreo en tiempo real:** La integración de comunicación en tiempo real con internet de las cosas (IoT, por sus siglas en inglés) y sistemas de supervisión, control y adquisición de datos (SCADA, por sus siglas en inglés) permite el monitoreo de maquinaria, proporcionando a los usuarios una visión integral e intuitiva de los procesos. En el estudio de [Marino*, et al.*, 2021], se presentan dos casos de uso donde se desarrollaron sistemas para facilitar el monitoreo del estado de maquinaria mediante *displays* montados en la cabeza (*headsets*). Estas soluciones no solo impactan positivamente en la operación, sino que también podrían tener un potencial impacto social significativo, pues podrían transformar a personas no especializadas en operadores altamente calificados y efectivos, permitiéndoles comunicarse y compartir su campo de visión con expertos en tiempo real para tomar decisiones más adecuadas de manera conjunta.

1.2.2. Implementaciones de AR en la industria

Uno de los sectores productivos que ha encontrado gran utilidad en la incorporación de experiencias de AR es el de la fabricación de automóviles. Según [Saldamando, 2024], director ejecutivo de *Center of Engineering and Advanced Technology* S.A. de C.V., mantener altos estándares de calidad es una obligación en este sector, y la AR ha demostrado ser una herramienta revolucionaria para los procesos operativos. Entre los ejemplos destacados de adopción de esta tecnología, menciona: en Porsche, donde los operadores pueden conectarse directamente con expertos de forma remota para resolver problemas; en BMW, donde se utilizan modelos 3D superpuestos sobre las carrocerías de los autos para verificar posibles imperfecciones; y en Ford, donde se emplea para la capacitación acelerada del personal, reduciendo errores de novato. Además, el director destacó que la herramienta contribuye a minimizar errores, reducir costos y asegurar altos estándares de calidad, mientras que el acceso a información y asistencia mejora la eficiencia en el piso de planta.

Una aplicación destacada en este ámbito es la planta piloto de BMW en Múnich, donde se emplea una aplicación de AR para validar desde secciones individuales de vehículos hasta etapas completas de producción, reduciendo el tiempo de validación y prototipado en casi doce meses. Esta aplicación está conectada al sistema de administración de datos de productos de BMW, que proporciona archivos de diseño asistido por computadora (CAD, por sus siglas en inglés) utilizados por los especialistas para verificar si los operadores podrán instalar correctamente los componentes. Este enfoque acelera la fase de pruebas de los procesos de producción, resultando en un ahorro significativo de tiempo y dinero al integrar nuevos vehículos en la línea de producción. La empresa ha reconocido el valor de esta herramienta y continúa trabajando en expandir sus capacidades, incorporando datos de IoT y reconocimiento inteligente de objetos [BMW Group, 2020].

Howden, una empresa especializada en soluciones para la industria de procesos, como ventilación minera, tratamiento de aguas residuales y sistemas de refrigeración, enfrentó la necesidad de adaptarse a un mercado competitivo. Para ello, adoptó una solución de AR utilizando Vuforia Studio y los Microsoft HoloLens. Este enfoque tenía como objetivo sustituir el servicio reactivo por una colaboración proactiva con sus clientes, proporcionando una herramienta para visualizar y comprender de manera más clara el rendimiento y estado de los equipos a través de datos IoT. La experiencia permitió a

los operadores obtener una visión inmersiva y detallada del funcionamiento interno de los equipos, facilitando el mantenimiento predictivo, la identificación rápida de piezas y la reparación [PTC, 2021].

1.2.3. Entornos de desarrollo de AR

Los entornos de desarrollo de experiencias de AR, conocidos como kit de desarrollo de software (SDK, por sus siglas en inglés), han experimentado una gran evolución desde la introducción de ARtoolkit en 1999 hasta la aparición de entornos como ARkit y ARCore, de Apple y Google respectivamente.

Actualmente, existen SDKs para áreas específicas, como AR Expeditions de Google para la educación de nivel básico [Kljun*, et al.*, 2020], Unity y Unreal Engine para robótica tanto social como industrial [Coronado*, et al.*, 2023] y Vuforia para experiencias educativas en el área STEM (*Science, Technology, Engineering, and Mathematics*) [GodoyJr, 2020].

De acuerdo a lo anterior, se puede inferir que la elección del SDK más adecuado para desarrollar una experiencia depende de las características específicas que se deseen de la aplicación y de las herramientas ofrecidas por cada *kit*. Estudios como "*A Comparative Analysis of Augmented Reality Frameworks Aimed at the Development of Educational Applications*" [Fabrício Herpich, 2017] son útiles para comparar las capacidades de diferentes soluciones en el mercado. En dicho trabajo se identificaron las características clave para el desarrollo de experiencias de AR y se establecieron criterios de evaluación en tres categorías: información general, identificación y seguimiento de marcadores, y funcionalidades adicionales. Estas incluyen subcategorías definidas para facilitar la comparación entre los 11 marcos de desarrollo analizados, los cuales se presentan en la Tabla 1.1:

General Information		**Target tracking types**							**Additional features**		
SDK	**Plataforma**	**Tutorial**	**Text**	**Planar Image**	**3D Object**	**Multi Targets**	**Geo-location**	**Markerless**	**Online recognition**	**Offline recognition**	**AR Editing Platform**
ARToolKit (5.3.2)	Windows, Mac OS, Linux, iOS, Android, Unity Package, Smart Glasses	Yes	No	Yes	No	Yes	Yes	No	No	Yes	No
Augment (3.2.1-1)	App (Android, iOS, Windows, Mac)	Yes	No	Yes	No	No	No	No	Yes	No	Yes
Aurasma (3.5.3)	App (Android e iOS)	Yes	No	Yes	No	No	Yes	No	Yes	No	Yes
BlippAR (2.1.1)	App (Android e iOS)	Yes	No	Yes	No	No	No	No	Yes	No	Yes
CraftAR (3.1.3)	App (Android, iOS), Unity Package, Apache Cordova	Yes	No	Yes	No	No	No	Yes	Yes	Yes	Yes
EasyAR (1.3.1)	Windows, Mac OS, Android, Unity Package	Yes	No	Yes	No	Yes	No	No	No	Yes	No
Kudan (1.5)	Android, iOS, Unity Package	Yes	No	Yes	No	Yes	No	No	No	Yes	No
LayAR (8.5.3)	App (Android, iOS e BlackBerry)	Yes	No	Yes	No	No	Yes	No	Yes	No	Yes
PixLive (5.6.0)	App (Android, iOS), Apache Cordova, Google Glass	Yes	No	Yes	No	No	Yes	No	Yes	No	Yes
Vuforia (6.2)	Windows, iOS, Android, Smart Glasses, Unity Package	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No
Wikitude (2.1.0-2.1.0)	App (Android e iOS), Unity Package, Cordova, Titanium, Xamarin, Smart Glasses	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes

Tabla 1.1 Comparación de características de plataformas de AR.

Según la Tabla 1.1, el SDK de Vuforia destaca por ofrecer una amplia variedad de herramientas de desarrollo, especialmente en el seguimiento de diferentes tipos de objetos, como imágenes planas,

objetos 3D y seguimiento sin marcadores. Asimismo, su enfoque hacia aplicaciones industriales, junto con su integración con sistemas existentes, interfaz intuitiva y herramientas especializadas, simplifica el desarrollo y despliegue de aplicaciones complejas, posicionando a Vuforia como una solución adecuada para explorar el uso de la AR en entornos industriales [Vuforia Engine Team, 2024].

1.3. Formulación del problema

Esta investigación se sitúa en el campo de la realidad aumentada (AR) aplicada a la industria, con énfasis en el área operativa, que abarca tareas como operación, supervisión, mantenimiento, ensamblaje y capacitación en instalaciones industriales [Moreno, 2001]. Aunque la automatización industrial es esencial para garantizar la eficiencia y seguridad en estos entornos [Sanchis Llopis*, et al.*, 2010], enfrenta desafíos significativos: la necesidad de personal con conocimientos sólidos [Escaño González*, et al.*, 2019], y la adaptación a los nuevos estándares de flexibilidad, resiliencia y calidad impulsados por la Industria 4.0 [Kamble*, et al.*, 2018].

Actualmente, los procedimientos de operación, mantenimiento y supervisión en instalaciones industriales presentan desafíos relacionados con la eficiencia y propensión a errores [Garza*, et al.*, 2013], debido a la complejidad tanto de los sistemas como de las tareas a realizar [Tortora*, et al.*, 2021], así como la necesidad de conocimientos especializados [Comisión Electrotécnica Internacional, 2010]. La falta de herramientas diseñadas para facilitar la ejecución y comprensión de estas tareas, junto con la dependencia de procedimientos y manuales escritos [Hou*, et al.*, 2013], [Larson, 2022], contribuyen a la ineficiencia operativa y aumentan el riesgo de errores, lo que puede conllevar costos elevados y situaciones peligrosas [Veinott y Kanki, 1995]. Por ello, es crucial implementar soluciones que asistan a los trabajadores en sus labores diarias y transmitan el conocimiento de manera más eficaz.

Atender esta problemática es prioritario por tres razones: primero, la ineficiencia en los procesos operativos puede resultar en tiempos de inactividad prolongados, afectando la producción y aumentando sus costos [Garza*, et al.*, 2013], [Papakostas*, et al.*, 2022], [Tortora*, et al.*, 2021]. Segundo, reducir los errores y mejorar la seguridad es crucial, ya que los fallos en la operación y el mantenimiento pueden comprometer la integridad del personal y los equipos [Longo*, et al.*, 2017], [Comisión Electrotécnica Internacional, 2010]. Finalmente, en el contexto de la Industria 4.0, donde se requieren sistemas flexibles y ágiles que se adapten rápidamente a los cambios, la ineficiencia operativa dificulta la consecución de estos objetivos, reduciendo la competitividad, especialmente frente a competidores que adopten nuevas tecnologías a sus procesos [Morales Méndez y del Cerro Velázquez, 2024].

La incorporación de AR ofrece una oportunidad para superar estos desafíos y posiciona a las empresas a la vanguardia tecnológica, mejorando su competitividad en el mercado. Esta tecnología es considerada clave en la transformación digital por empresas líderes en el mercado [Rockwell Automation, 2023]. De la misma forma, Gartner la considera una tecnología madura y lista para incorporarse a procesos productivos [Herdina, 2020]. Además, las investigaciones centradas en tendencias de manufactura la colocan como una herramienta clave, ya que puede proporcionar procedimientos innovadores con el potencial de mejorar la eficiencia operativa y reducir los errores [De Pace*, et al.*, 2018], [Tortora*, et al.*, 2021], al ofrecer una forma más interactiva y comprensible de mostrar información [Thees*, et al.*, 2020], [Mayer y Fiorella, 2014] acerca de la realización de tareas, supervisión de sistemas y capacitación del personal en instalaciones industriales.

Para incorporar esta tecnología a procesos reales, es crucial considerar herramientas especializadas que permitan desarrollar aplicaciones de AR específicas para entornos industriales, que ofrezcan integración con sistemas, facilidad de uso, soporte técnico y herramientas orientadas a la industria. Por lo tanto, el uso de una plataforma como Vuforia Studio se justifica por su especialización en tecnología industrial, su enfoque en Industria 4.0 y la gran variedad de herramientas para crear experiencias de alto impacto [Vuforia Engine Team, 2024], [PTC, 2019].

1.4. Objetivo

El objetivo principal de este trabajo es desarrollar y validar una experiencia de AR aplicada a una instalación industrial de manejo de líquidos, propiedad del laboratorio de automatización de la Facultad de Ingeniería de la Universidad Nacional Autónoma de México. El sistema, compuesto por dos tanques atmosféricos interconectados, comparte características y desafíos con instalaciones industriales de procesos productivos. La integración de AR en este entorno busca optimizar las tareas operativas, promover la adopción de tecnologías emergentes y demostrar el valor de esta tecnología en contextos industriales.

Para asegurar estos beneficios e impacto, la aplicación debe cumplir cierto estándar mínimo de rendimiento; por lo tanto, se llevará a cabo una evaluación técnica de la aplicación, enfocada en el consumo de recursos de computación, tiempos de respuesta, usabilidad y calidad visual.

1.4.1. Objetivos particulares

1. Desarrollar una experiencia de AR para el sistema de tanques atmosféricos interconectados.
2. Utilizar herramientas enfocadas al área industrial, específicamente el software Vuforia Studio.
3. Permitir a los operadores visualizar descripciones, mostrar animaciones de ensamblaje, funcionamiento y vistas explosionadas de los equipos que componen el sistema, a través de la pantalla de un dispositivo móvil.
4. Añadir interactividad a la experiencia, incluyendo controles para animaciones y secuencias de pasos, botones de navegación y respuestas a gestos realizados con los dedos sobre la pantalla.
5. Dotar de accesibilidad a la experiencia, por lo cual esta debe ser capaz de ejecutarse de forma local en múltiples dispositivos y sistemas operativos.
6. Evaluar métricas de rendimiento, como tasa de fotogramas por segundo (FPS, por sus siglas en inglés), latencia de la interacción, uso de recursos de memoria (RAM, GPU y CPU), compatibilidad, calidad visual y consumo de datos de red.

1.5. Contribuciones

- Aplicación multiplataforma desarrollada en Vuforia Studio, desplegable desde Vuforia View que contiene la experiencia de AR en una interfaz de usuario accesible e intuitiva.
- Animaciones de funcionamiento y puesta en marcha, así como acceso de información detallada de los elementos que conforman el sistema de tanques atmosféricos interconectados.
- Trabajo escrito que detalla los conceptos teóricos necesarios y el proceso de creación de experiencias en Vuforia Studio.

1.6. Organización de la tesis

El presente trabajo de tesis se encuentra dividido en cinco capítulos, siendo el presente el que concierne a la introducción. Los siguientes cuatro se describen a continuación:

En el **Capítulo 2**, se revisa el marco teórico, partiendo de lo general a lo particular. Primero, se exploran los conceptos fundamentales y el funcionamiento de la AR. A continuación, se aborda la Industria 4.0 y la digitalización industrial. Posteriormente, se examinan las tecnologías CAD, sus características técnicas como tipos de archivo, *software* de diseño e integración con la AR; además, se describe la relación de estas tecnologías con la Industria 4.0. Las siguientes secciones detallan los conceptos teóricos necesarios para comprender el funcionamiento de Vuforia Studio, incluyendo sus funcionalidades, las características que lo hacen adecuado para conseguir los objetivos planteados y el entorno de *software* propuesto por PTC. También se profundiza en los conceptos de programación necesarios para crear experiencias en este entorno de desarrollo. Finalmente, se describe la construcción y funcionamiento del sistema de tanques atmosféricos interconectados, que es la instalación industrial sobre la que será implementada la experiencia desarrollada.

En el **Capítulo 3**, se detalla el proceso de creación de la experiencia, comenzando con la preparación y optimización de los modelos CAD, seguida del proceso de importación al proyecto y metodologías seguidas para usar estos recursos en la experiencia. Posteriormente, se describe la utilización de hojas de estilo en cascada (CSS, por sus siglas en inglés) para crear los aspectos gráficos y definir el posicionamiento y escala de los *widgets*, haciendo énfasis en la creación y asignación de clases y la implementación de diseño responsivo. Después, se describe el uso de eventos en Vuforia Studio, utilizados para crear botones de navegación, desplegar y ocultar elementos emergentes, así como ejecutar funciones y secuencias de animaciones. En este último apartado, también se detalla la estructura lógica de las secuencias en JavaScript. Posteriormente, se explica el funcionamiento de las animaciones generales utilizadas para crear secuencias complejas, detallando su lógica de programación. Finalmente, se mencionan las prácticas y criterios seguidos para utilizar los *widgets* de seguimiento y se describe el proceso de publicación de la experiencia.

En el **Capítulo 4**, se lleva a cabo la evaluación técnica de la aplicación, donde se presentan las métricas necesarias para analizar el desempeño de la experiencia en plataformas Windows y Android. Cada subsección detalla la metodología empleada para obtener las mediciones en cada uno de los tres dispositivos utilizados y se exponen las mediciones obtenidas. Posteriormente, se sintetizan los resultados y se realiza un análisis de los datos, evaluando el rendimiento y la usabilidad de la aplicación, así como las consideraciones clave para lograr un rendimiento óptimo.

Finalmente, en el **Capítulo 5**, se presentan las conclusiones del desarrollo, destacando los resultados obtenidos y evaluando si se logró cumplir con el objetivo planteado. Asimismo, se discute el impacto esperado de la investigación, se describen las limitaciones del estudio y se ofrecen sugerencias para futuros desarrollos en el área.

Capítulo 2

Marco Teórico

2.1. Conceptualización y componentes de la AR

En la introducción se abordaron las características generales de la realidad aumentada, definiéndola como una tecnología que puede aportar valor en el ámbito operativo de instalaciones industriales. Sin embargo, para comprender su aplicación en esta área, es necesario profundizar en conceptos más avanzados, para lo cual se analizarán los componentes clave de la AR para entender cómo pueden aplicarse y utilizarse eficazmente para contribuir a crear experiencias con impacto en dichos entornos.

Primero, se definirán los componentes esenciales de una experiencia de AR. Según [Telefónica, 2011], se requieren cuatro elementos básicos:

1. Un componente para capturar las imágenes de la realidad, generalmente una cámara, presente en los dispositivos móviles actuales.

2. Un elemento sobre el cual proyectar la superposición de datos virtuales con imágenes reales, generalmente la pantalla del dispositivo aunque, como vimos en la introducción, también puede ser un banco de trabajo u otros medios.

3. Un elemento de procesamiento, el cual se encarga de interpretar la información del mundo real, generar la información virtual y mezclarla de forma adecuada para ser proyectada.

4. Finalmente, se requiere un elemento denominado *trigger*, o activador de la experiencia. Este puede ser una imagen, coordenadas de localización mediante sistema de posicionamiento global (GPS, por sus siglas en inglés), códigos bidimensionales o marcadores. Una vez que el dispositivo reconoce este elemento, se despliega la experiencia de AR.

La AR es una variante de los entornos virtuales, comúnmente conocidos como VR. Estas tecnologías sumergen completamente al usuario en un entorno artificial, bloqueando la percepción del mundo real. En contraste, la AR permite al usuario ver el mundo real mientras superpone o combina objetos virtuales con el entorno físico; así, en lugar de reemplazar la realidad, la AR la complementa [Azuma, 1997]. Para analizar estas diferencias clave se elaboró la Tabla 2.1.

2.1.1. Superposiciones digitales (*digital overlays*)

La ejecución de una experiencia de AR se puede dividir en dos procesos, que suceden consecutivamente en pasos de tiempo de forma cíclica; [Craig, 2013] los define como:

Característica	Realidad Aumentada (AR)	Realidad Virtual (VR)
Interacción Ambiental	Integra elementos digitales en el mundo real, permitiendo interacción simultánea con ambos.	Crea un entorno completamente virtual, desconectado del mundo físico.
Percepción del Usuario	El usuario permanece en contacto con el mundo real, viéndolo a través de una superposición digital.	El usuario está completamente inmerso en un entorno virtual, perdiendo contacto con el mundo real.
Requisitos de *Hardware*	Generalmente requiere un *smartphone* o *tablet* para su uso.	Requiere equipos especializados como cascos, guantes, y en ocasiones, configuraciones de habitación a escala.
Aplicaciones Comunes	Juegos que proyectan elementos digitales en el entorno real, visualización de muebles en un espacio real antes de comprarlos.	Juegos inmersivos, simulaciones de entrenamiento para pilotos, cirujanos, entre otros.

Tabla 2.1 Comparativa entre realidad aumentada (AR) y realidad virtual (VR).

1. La aplicación debe procesar las condiciones actuales, tanto del mundo físico como del virtual.
2. La aplicación debe reflejar los datos virtuales superpuestos en el mundo real, de manera que los participantes perciban los primeros como parte de su entorno físico, para luego regresar al paso 1 y avanzar al siguiente paso de tiempo.

El segundo paso está relacionado con el concepto de superposiciones digitales, también conocidas como *digital overlays*; este término se refiere a las capas de información digital que se colocan sobre el mundo físico. El contenido puede incluir imágenes, sonidos, videos, gráficos o datos GPS.

En el contexto industrial, la superposición de información en AR se utiliza para aumentar las capacidades de los operadores y técnicos, asistiendo en la realización de tareas complejas para ser realizadas con mayor precisión y eficacia, esto mediante el despliegue de información relevante acerca de la tarea o procedimiento en curso directamente sobre el campo visual del profesional. Esta información puede incluir: secuencias de pasos, botones para avanzar en las etapas, barras de estado, tarjetas de información, advertencias visuales para zonas de riesgo o peligros en la operación, animaciones que lo guían en la correcta ejecución de procedimientos, entre otras.

2.1.2. Interactividad

La interactividad es una característica omnipresente en la mayoría de las aplicaciones móviles actuales, donde la interacción directa del usuario es esencial para desplegar información en la pantalla de los dispositivos. Una de las principales cualidades de la AR es la capacidad de ser percibida y utilizada de manera más interactiva en comparación con otros medios convencionales. De la misma forma en que se mira una película o se lee un libro, se puede decir que la AR se experimenta [Craig, 2013].

En este tipo de experiencia, se enlazan las interacciones del usuario, como gestos, comandos de voz, contacto y movimientos sobre la pantalla, entre otros, con la superposición de contenido digital sobre el mundo físico, habilitando a los usuarios para manipular objetos digitales y recibir realimentación inmediata [Azuma, 1997].

En el contexto industrial, la interactividad ofrecida por la AR se utiliza para tareas de mantenimiento predictivo, capacitación y resolución de problemas. Esta herramienta habilita a los profesionales para simular procesos o interactuar virtualmente con componentes de maquinaria antes de cualquier

intervención física, reduciendo el riesgo de cometer errores o provocar daños tanto a sí mismos como a los equipos [Craig, 2013].

Una aplicación que ilustra esta característica y su capacidad para eficientar tareas es el caso de la empresa logística DHL, la cual ha implementado AR en sus almacenes para asistir a los trabajadores. Estos usan lentes inteligentes que les muestran la ubicación de artículos a empacar, la ruta más rápida para llegar a ellos y el lugar donde deben ser colocados en el carro [Mendoza-Ramírez*, et al.*, 2023]. En este caso se destaca cómo la AR sobrepasa las capacidades de interacción del trabajador respecto a otros medios, mejorando la eficiencia de la tarea.

2.1.3. Registro espacial y seguimiento (*spatial registration and tracking*)

Una experiencia de AR comienza con la captura de señales del mundo real, como video y audio, las cuales son procesadas para mejorar la imagen, lo que permite la segmentación de objetos y el reconocimiento de patrones. A partir de este proceso, se determina qué objetos reales deben ser reemplazados por virtuales y dónde deben colocarse en la representación del espacio físico, considerando la posición y perspectiva adecuada [Redondo, 2012].

Posteriormente, se realiza el registro espacial, es decir, la alineación precisa de los objetos virtuales con el mundo real, asegurando que se mantengan en la misma posición y orientación relativas al entorno físico, incluso cuando el usuario se mueve. Por ejemplo, si la meta de una experiencia es poner unos lentes de sol en el rostro de alguien, el registro espacial se encarga de posicionar la representación tridimensional de los lentes siempre sobre el puente de la nariz y en la orientación correcta [Craig, 2013].

Dado que las representaciones de la realidad desplegadas por la AR deben ser consistentes con el entorno en todo momento, surge el problema del seguimiento (*tracking* por su traducción al inglés), el cual se refiere al monitoreo continuo de la posición y orientación del dispositivo en relación con el entorno. Para conseguir un registro espacial exacto, se utilizan seis grados de libertad, que incluyen las posiciones en los ejes `X`, `Y` y `Z`, para establecer la ubicación y los ángulos de rotación existentes alrededor de dichos ejes para definir la orientación [Craig, 2013].

La experiencia debe tener datos del mundo físico en tiempo real, específicamente la estimación de la orientación y posición del usuario para deducir qué está observando. La obtención de estos datos se realiza mediante algunos métodos, que según [Craig, 2013] y [Redondo, 2012] incluyen, pero no se limitan, a los siguientes:

- **Sensores usados para seguimiento:** Pueden ser pasivos, los cuales no requieren de la interacción del usuario para conseguir los datos (giroscopios, sensores de campo eléctrico, acelerómetros), y activos, como botones, teclados y actuadores.
- **Cámaras (visión por computadora):** Utiliza una cámara como entrada de información, establece la ubicación y orientación del participante en términos absolutos (coordenadas específicas) o de forma relativa (respecto a un objeto del entorno). Además, establece la posición y orientación de la cámara basándose en puntos de referencia, que pueden ser características naturales del entorno o marcadores fiduciales, tales como códigos *quick response* (QR, por sus siglas en inglés). Cuando el sistema detecta el marcador, ya sea natural o artificial, despliega los datos virtuales en la posición correcta. En la Figura 2.1 se muestra un ejemplo de seguimiento mediante este método, usando marcas fiduciales.

- **GPS:** Es un sistema de navegación con la capacidad de determinar la localización en coordenadas `X` e `Y` del usuario. En dispositivos móviles, esto se complementa con una brújula digital que permite conocer la orientación del usuario, simplificando los cálculos necesarios para el despliegue de la información.

En las aplicaciones diseñadas para instalaciones industriales, es crucial realizar un seguimiento preciso de los objetos aumentados, pues la capacidad de optimizar las operaciones y facilitar las tareas depende de que la información digital se despliegue en el momento y lugar adecuados, dentro del campo de visión del trabajador y sin representar una distracción o un obstáculo. En el caso de las animaciones sobrepuestas a los equipos, el seguimiento debe ser lo suficientemente preciso para generar una experiencia inmersiva que permita a los trabajadores mejorar la adquisición de conocimiento y la ejecución de sus tareas. Para las aplicaciones de revisión de calidad, el seguimiento debe ser exacto para que el operador pueda detectar inmediatamente cualquier imperfección en la superficie del objeto.

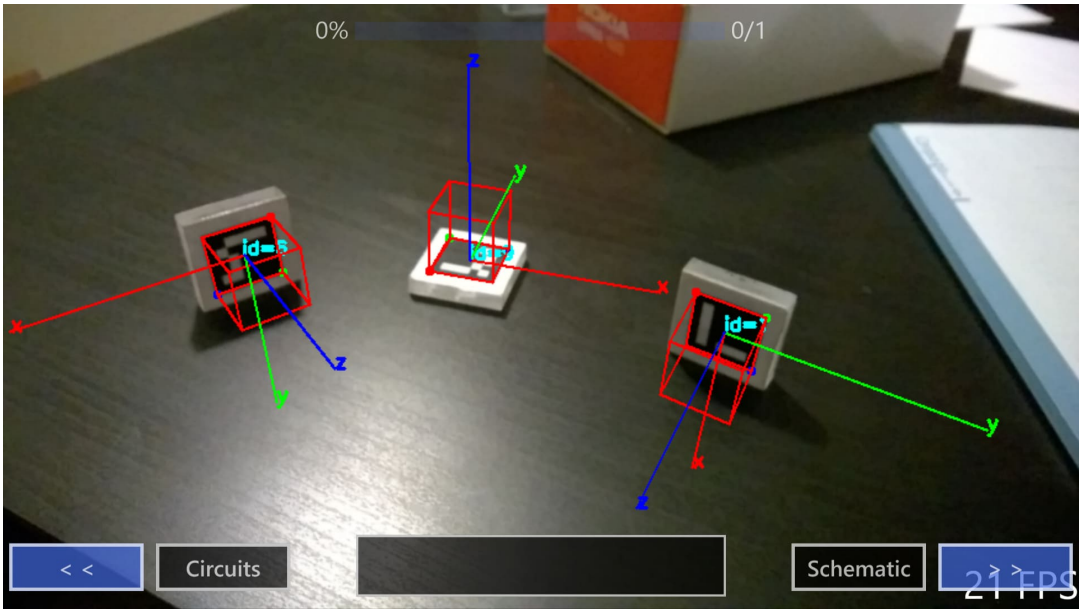


Figura 2.1 Seguimiento a través de visión por computadora de objetos con distintas orientaciones. [1]

2.1.4. Interfaz de usuario (User Interface, UI)

En la AR, el campo de visión del usuario es de vital importancia porque este medio es el predominante para presentar la información digital, también conocida como *digital overlay*. Estas superposiciones pueden incluir datos tridimensionales (como animaciones, textos flotantes y modelos 3D), así como datos bidimensionales (botones, tarjetas de información, imágenes, entre otros). La segunda categoría de datos puede ser agrupada bajo el concepto de interfaz de usuario (UI, por sus siglas en inglés).

La UI es un concepto que abarca arquitectura de información, patrones y elementos visuales que facilitan la interacción eficiente con sistemas operativos y *software* en diversos dispositivos; en el contexto de la AR, funciona como el medio a través del cual el usuario interactúa con una experiencia [Corrales, 2019].

[1] Obtenido de: [Yngchen, 2014].

El paradigma tradicional de interfaz de usuario basado en ventanas, íconos, menús y punteros (WIMP) no se adapta bien a los sistemas de AR. A diferencia de las interfaces 2D, en estas experiencias se requiere interacción en seis grados de libertad (6DOF), y el uso de dispositivos convencionales como el ratón y el teclado resulta incómodo y limita la usabilidad. Al igual que en las interfaces WIMP, las interfaces de AR deben permitir la selección, el posicionamiento y la rotación de objetos virtuales, así como la creación de trayectorias, la asignación de valores cuantitativos y la entrada de texto. Sin embargo, una diferencia clave en la interacción de la AR es que también incluye la selección, anotación y, en algunos casos, la manipulación directa de objetos físicos [Van Krevelen y Poelman, 2010].

La UI, en el contexto industrial, se utiliza para desplegar las funcionalidades que asisten al profesional en el cumplimiento de sus tareas, por lo cual es fundamental garantizar la disponibilidad y accesibilidad de estas herramientas en todo momento [Duda*, et al.*, 2022], así como permitirle realizar y deshacer acciones dentro de la experiencia [Corrales, 2019]. De la misma forma, la interfaz debe transmitir la información de manera precisa, mostrando únicamente los datos relevantes para el usuario y siendo visualmente atractiva para facilitar la creación de patrones de uso.

2.1.5. Conectividad y computación en la nube

El concepto de computación en la nube se refiere a la entrega de servicios relacionados con las tecnologías de la información (IT, por sus siglas en inglés), como servidores, almacenamiento o bases de datos a través de internet. Se denominan "nube" debido a que estos servicios son gestionados de forma remota por un tercero, eliminando la necesidad de infraestructura física en la ubicación del usuario. Entre las ventajas de la computación en la nube (o cloud computing) destacan la reducción de costos operativos, la agilidad en la prestación del servicio, la escalabilidad y una accesibilidad global independiente de la ubicación geográfica del usuario y del proveedor [Kamble*, et al.*, 2018].

Como se mencionó en la sección de antecedentes, la AR está estrechamente vinculada con la conectividad y el *cloud computing*. La combinación de estas tecnologías habilita nuevas formas de compartir experiencias e intercambiar información con expertos, fabricantes o bases de datos, lo que abre la puerta a nuevas formas de negocio, como el modelo sistema de producto-servicio (PSS, por sus siglas en inglés) [Mourtzis*, et al.*, 2017]. Además, la conectividad en la nube es fundamental en el contexto de la AR, ya que permite el acceso y procesamiento de grandes volúmenes de datos en tiempo real, lo que también mejora la colaboración remota, como se evidencia en el estudio de [Mourtzis*, et al.*, 2017].

Además, según [Chen*, et al.*, 2019] la implementación de la conectividad de experiencias de AR con la nube, sirve para abordar ciertos desafíos, como los relacionados con el almacenamiento y la demora en el intercambio de información, entre otros. Algunas formas de interacción aún no están completamente desarrolladas, como los comandos de voz, los cuales requieren tecnología de inteligencia artificial (IA, por sus siglas en inglés), así como mejorar la conectividad con dispositivos IoT para proporcionar información más detallada en las experiencias.

Estos desarrollos pueden permitir que las experiencias AR estén conectadas no solo con servicios de computación en la nube, sino también con tecnologías como IoT y conectividad 5G, para obtener información directamente de internet en el campo de visión del profesional en momentos relevantes. Algunas posibilidades de conectividad se encuentran en la Figura 2.2.

Uno de los usos aún por explorar de esta integración es el potencial de transferir tareas computacionales más pesadas a la nube. Esto permitiría a dispositivos con recursos limitados ejecutar experiencias

[2]Obtenido de: [Raza, 2019].

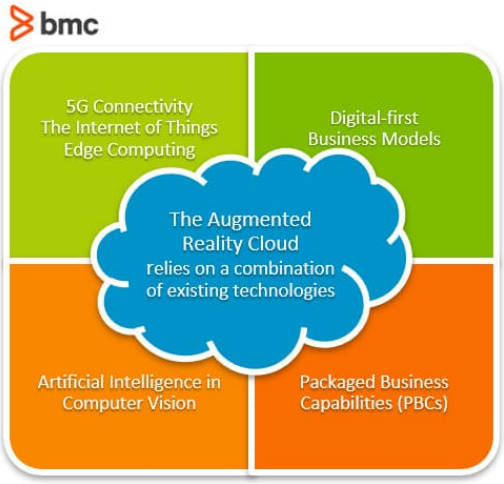


Figura 2.2 Conectividad de la AR con diversas tecnologías. [2]

de AR y aumentaría las capacidades de otros dispositivos para reproducir experiencias más complejas o de mayor precisión. Un ejemplo sería mejorar las capacidades de visión por computadora de un sistema, procesando los datos mediante servicios de *cloud computing* y desplegando el resultado de manera más eficiente en la experiencia [Raza, 2019]. Además, esta integración facilitaría la colaboración sin interrupciones, el intercambio de contenido y el uso de aplicaciones que requieren gran poder computacional, sin sobrecargar los dispositivos locales [Mendoza-Ramírez*, et al.*, 2023].

2.1.6. *Hardware* y dispositivos de visualización

Todas las herramientas físicas necesarias para desplegar una experiencia de AR se denominan *hardware*, éste puede incluir dispositivos de visualización, también conocidos como *displays* (celulares, tabletas electrónicas, pantallas montadas en la cabeza), dispositivos de entrada de información para la experiencia o sensores (teclados, micrófonos, giroscopios, acelerómetros, botones físicos), y procesadores, cuya función es realizar la computación necesaria para ejecutar las experiencias y conectarlas con otros servicios de IT (computadoras, servidores). Un ejemplo que integra todos estos elementos se encuentra en el trabajo de [Mourtzis*, et al.*, 2017], mencionado en los antecedentes de esta investigación.

En general, el *hardware* necesario para ejecutar y conectar las experiencias ya está establecido en el mercado, sin variaciones significativas. Además, para el objetivo de este desarrollo, se considera que la vista es el medio predominante para percibir las experiencias de AR; por lo tanto, únicamente se analizarán a fondo los medios de visualización.

Los *displays* se pueden definir como las herramientas tecnológicas que integran información virtual en el entorno real, amplificando las percepciones visuales, auditivas y táctiles. Estos dispositivos se agrupan en varias categorías, lo que permite una clasificación práctica de la tecnología disponible actualmente: (1) cascos y gafas inteligentes, (2) dispositivos portátiles, (3) sistemas basados en proyección, (4) realidad aumentada espacial (SAR), (5) HUD (pantallas de visualización frontal) y (6) lentes de contacto de AR [Mendoza-Ramírez*, et al.*, 2023]. La diversidad de dispositivos se ilustra en el esquema de la Figura 2.3.

Los *displays* de AR de interés para esta investigación son las *smart glasses*, *headsets* y dispositivos portátiles. En el caso de los teléfonos inteligentes [Google, 2024] proporciona una lista de dispositivos

[3] Obtenido de: [Mendoza-Ramírez*, et al.*, 2023].

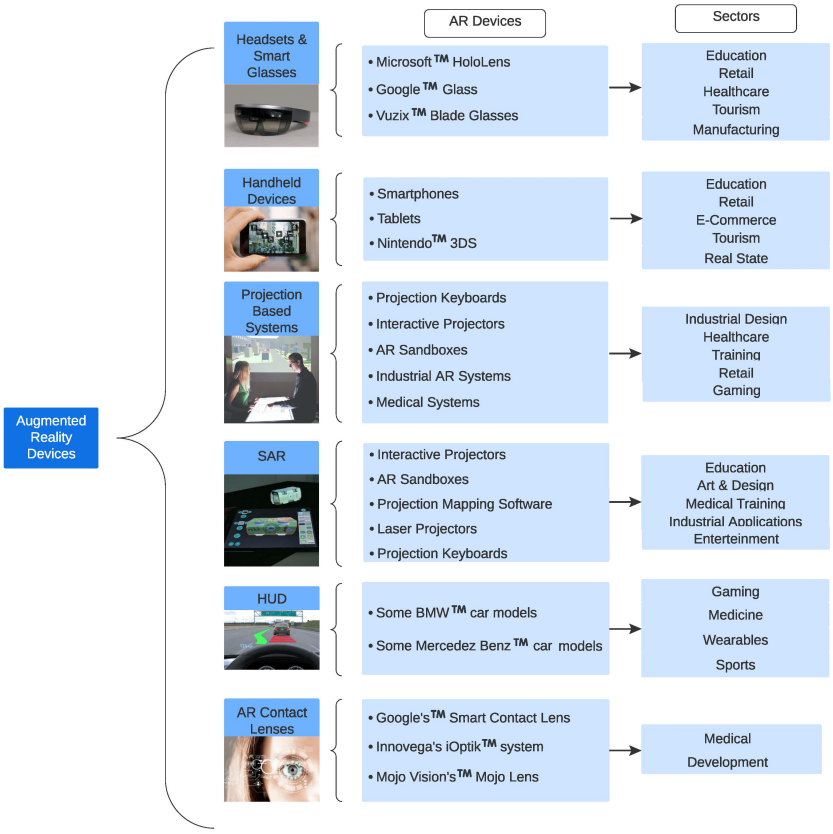


Figura 2.3 Tipos de dispositivos de visualización de AR. [3]

compatibles con ARCore, el motor que habilita la ejecución de experiencias. De manera general, cualquier dispositivo móvil que cuente con las siguientes características específicas tiene la capacidad de ejecutar una experiencia de forma fluida y efectiva:

1. Sistema operativo Android 7.0 o superior, o iOS 11 o superior.
2. Procesadores Qualcomm Snapdragon 835 o superiores, en iOS: Procesadores A9 o superiores (iPhone 6s y modelos posteriores).
3. Al menos 3 GB de memoria RAM.
4. Giroscopio y acelerómetro.
5. Procesador gráfico (GPU) compatible con OpenGL 3.0.
6. Cámaras con capacidad de enfoque automático.

En las categorías de *headsets* y *smart glasses*, la mayoría de dispositivos integran varios componentes clave: en primer lugar, un *display* generalmente transparente para permitir al usuario observar tanto el mundo real como los elementos virtuales superpuestos. En segundo lugar, un procesador para ejecutar y desplegar la experiencia. Finalmente, estos dispositivos incluyen sensores y cámaras, como sensores de profundidad y cámaras para el reconocimiento de imágenes y el seguimiento de objetos físicos. Además, algunos de los *headsets* actuales integran sistemas de transmisión de sonido, micrófonos

para capturar la voz, así como otros elementos de entrada que permiten al usuario interactuar con la experiencia de manera inmersiva.

Para presentar y comparar algunas de las opciones más populares existentes en el mercado, se elaboró la Tabla 2.2 con información tomada de páginas *web* especializadas; [Microsoft, 2024], [Hector, 2023], [Lenovo, 2020], [Magic Leap, 2024], [Heaney, 2024].

Headset	**Resolución**	**Campo de Visión (FOV)**	**Peso**	**Procesador**	**Batería**	**Ventajas**	**Desventajas**
Meta Quest 3	2064 x 2208 píxeles por ojo	110 grados	515 g	Qualcomm Snapdragon XR2+	2 horas 12 minutos	Gran mejora en gráficos y comodidad en comparación con su predecesor.	Duración limitada de la batería, falta de accesorios incluidos.
Microsoft HoloLens 2	2K por ojo	52 grados	566 g	Qualcomm Snapdragon 850	2-3 horas	Interfaz intuitiva, gestos naturales y soporte empresarial robusto.	Precio elevado, campo de visión limitado en comparación con otros modelos.
Lenovo ThinkReality A3	1080p por ojo	No especificado	<130g	Qualcomm Snapdragon XR1	6 horas (con batería externa)	Diseño modular y adaptabilidad para entornos empresariales.	Pesado en la nariz, caro y no muy cómodo para largos períodos de uso.
Magic Leap 2	1440 x 1760 píxeles por ojo	70 grados	260 g	AMD Zen 2 + RX 6000	3.5 horas	Excelente calidad de imagen y un diseño liviano.	Ecosistema limitado, software menos desarrollado que el de la competencia.
Apple Vision Pro	3660 x 3200 píxeles por ojo	aprox 100 grados	600-650 g	Apple M2 (5nm)	2-2.5 horas	Pantalla de alta definición, seguimiento ocular y manual avanzado, integración con el ecosistema Apple	Pesado, precio elevado, duración de la batería.

Tabla 2.2 Comparativa de *headsets* de AR.

En el contexto industrial, se busca que el *display* utilizado permita al usuario moverse con libertad, con especial énfasis en la capacidad de manipular objetos con ambas manos y desplazarse sin restricciones, mientras disfruta de una interfaz altamente funcional. Por esta razón, la combinación de *smart glasses*, *headsets* y dispositivos portátiles se ha consolidado como una de las opciones más populares en la industria [Mourtzis*, et al.*, 2017]. Además, es importante que estos dispositivos sean lo suficientemente robustos para operar en entornos industriales exigentes, soportando condiciones como temperaturas variables, abrasión, contaminación, entre otras. También deben garantizar una experiencia de usuario cómoda durante largas jornadas de trabajo, y proporcionar las capacidades necesarias para ejecutar aplicaciones de alta demanda.

2.2. Industria 4.0 y digitalización

2.2.1. Conceptualización de la Industria 4.0

Los paradigmas de fabricación han evolucionado para adaptarse a las necesidades del mercado, generando revoluciones industriales que incorporan nuevas tecnologías, métodos de trabajo y modelos de negocio. La cronología de esta evolución se puede ver en la representación de la Figura 2.4.

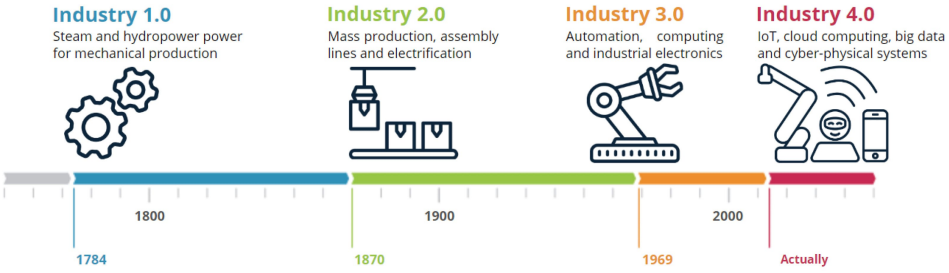


Figura 2.4 Línea del tiempo de la evolución industrial. [4]

La Industria 4.0 busca dar solución al deterioro económico, medioambiental y social mediante el uso de herramientas tecnológicas [Kamble, *et al.*, 2018]. En ese sentido, [Rockwell Automation, 2023] propone digitalizar la industria para hacer frente a retos comunes de los fabricantes, como la escasez de mano de obra calificada, la volatilidad del mercado, el incremento de los costos, las variaciones en las preferencias de los consumidores, las interrupciones en la cadena de suministro, la reducción de los tiempos de operación, los riesgos asociados a la ciberseguridad y las exigencias en materia de sostenibilidad.

Este nuevo paradigma industrial se caracteriza por la integración de técnicas y tecnologías avanzadas de operación, como ciberseguridad, AR, automatización, robótica, integración de sistemas, simulación, análisis de *big data*, manufactura aditiva, computación en la nube y el IoT [Morales Méndez y del Cerro Velázquez, 2024], con las tecnologías de fabricación, como CNC, tornos, fresadoras, amoladoras, robots actuadores, entre otras.

Las interacciones entre IT en las fábricas inteligentes se sustentan en tres componentes principales: sistemas ciberfísicos (*cyber-physical systems*, CPS), *cloud computing* y el IoT. Los CPS integran procesos de fabricación con sistemas computacionales para monitorear y ajustar las condiciones operativas en tiempo real, mientras que la computación en la nube facilita el almacenamiento, acceso y procesamiento eficiente de grandes volúmenes de datos. Por su parte, el IoT permite la interconexión entre dispositivos y sistemas, posibilitando la recopilación, transmisión y análisis de información que habilita funcionalidades avanzadas como mantenimiento predictivo, monitoreo remoto y optimización de procesos industriales [Kamble, *et al.*, 2018, IBM, 2024]. Como se puede inferir, integrar todo este flujo de datos y poder computacional con tecnologías AR aumenta aún más el impacto de esta tecnología en entornos productivos.

2.2.2. Relación de la Industria 4.0 con la AR

Las tecnologías disruptivas juegan un papel clave en la Industria 4.0, pues pretenden proporcionar herramientas para hacer frente a los retos de los fabricantes de manera más eficiente e innovadora,

[4] Obtenido de: [Morales Méndez y del Cerro Velázquez, 2024].

mejorando la colaboración, optimizando los recursos y promoviendo la sostenibilidad en los procesos industriales. La incorporación, tanto de estas tecnologías como de otras más establecidas en el mercado, se denomina transformación digital [Rockwell Automation, 2023]; en la Figura 1.2 se mencionan algunas de las tecnologías clave de esta tendencia, dentro de las cuales se encuentra la AR.

La integración de esta herramienta dentro de los procesos de la Industria 4.0 se puede clasificar en dos áreas: colaboración humano-máquina (*human-machine collaboration*) e integración de equipos en el piso de planta (*shop-floor equipment integration*) [Kamble*, et al.*, 2018]. Dicha integración mejora diversos aspectos de los procesos industriales y contribuye a la implementación de los principios de la Industria 4.0, como se detalla en la Tabla 2.3.

Principio	Descripción	Papel de la AR
Interoperabilidad	Capacidad para realizar la misma función al intercambiar máquinas y equipos, independientemente de sus fabricantes.	Permite a los operarios capacitarse rápidamente en el uso de nuevas máquinas.
Descentralización	Capacidad para que las decisiones se tomen de manera autónoma, por cada entidad que conforma al sistema.	Proporciona a los trabajadores acceso a información en tiempo real, lo que los habilita para tomar decisiones autónomas e informadas, la capacidad de compartir su campo de visión les permite ser asistidos por expertos de forma inmediata.
Virtualización	Creación de copias virtuales del mundo físico, enlazadas a datos de sensores para monitoreo y simulación.	Facilita la visualización de gemelos digitales de los equipos en la planta mediante AR. Los datos reales pueden enlazarse a las experiencias, creando simulaciones y monitoreo en tiempo real.
Capacidad en tiempo real	Capacidad para procesar y analizar datos en tiempo real, que transforma la toma de decisiones.	Se pueden mostrar datos en tiempo real, ayudando a los operarios a responder eficientemente, oportunamente y en el sitio.
Modularidad	Flexibilidad para adaptar sistemas a diferentes necesidades de producción mediante módulos intercambiables.	Facilita a los operadores las tareas de configurar, actualizar y ajustar módulos de maquinaria mediante instrucciones visuales.
Orientación al servicio	La flexibilidad y agilidad permiten a las empresas adaptarse a las necesidades cambiantes del mercado, facilitando el modelo PSS.	Permite soluciones de mantenimiento, asesoramiento, capacitación y servicios remotos por parte de los fabricantes, crea nuevas formas de vender.

Tabla 2.3 Integración de la AR con los principios de la Industria 4.0.

Algunas aplicaciones puntuales, además de las ya mencionadas, de la AR en la Industria 4.0 pueden ser la colaboración humano-robot, mediante la creación de interfaces eficientes para interactuar con robots industriales. En las tareas de mantenimiento-montaje-reparación, puede mejorar la productividad. En las operaciones de capacitación, mejora la adquisición de conocimientos. En la inspección de productos, los operadores pueden detectar cualquier discrepancia en los artículos inmediatamente usando primordialmente la vista [De Pace*, et al.*, 2018].

2.3. Diseño paramétrico y modelos CAD

2.3.1. Definición de diseño paramétrico

Un modelo es una representación que posee algunas o todas las características de algún fenómeno u objeto, ya sea real o abstracto [Ramirez, 1992]. En ingeniería, el uso de herramientas de CAD permite la creación de modelos digitales tridimensionales de objetos, piezas y maquinaria, los cuales se caracterizan por tener gran precisión geométrica.

Desde su aparición en los años 70, el CAD ha revolucionado el diseño y la manufactura, transformándolos en procesos ágiles, rápidos y precisos, pues ha posibilitado la creación de prototipos digitales inteligentes, en los que es posible realizar análisis físicos, simulaciones de comportamiento, planos muy detallados de piezas complejas, entre otras aplicaciones [Ramirez, 1992]. En la actualidad, la integración de estos modelos con otras tecnologías ha abierto la posibilidad a aplicaciones que transcienden el diseño, generando nuevas propuestas de valor para los productos y transformando la forma en que se interactúa con ellos; dentro de estas integraciones se encuentra la AR.

La alta precisión alcanzada por las herramientas CAD actuales permite que los modelos tridimensionales, que suelen ser el elemento predominante de los *digital overlays*, representen fielmente las herramientas y máquinas industriales, lo que facilita la creación de experiencias relevantes, inmersivas y efectivas para los profesionales. Por ello es crucial analizar más a fondo esta herramienta, así como su influencia y relación tanto con la AR como con la Industria 4.0.

Para realizar las representaciones tridimensionales, las primeras herramientas CAD utilizaban geometrías básicas, coordenadas vectoriales y operaciones booleanas; sin embargo, este enfoque generaba trabajo adicional y costos si se deseaban realizar modificaciones [Erazo-Arteaga, 2022]. Con el fin de facilitar esta labor, surge el diseño paramétrico, un enfoque dentro de las tecnologías CAD, el cual propone la realización del modelado bajo ciertos criterios geométricos que permitieran realizar distintas configuraciones al cambiar los valores de algunos parámetros, dando origen a herramientas como CATIA V5, SolidEdge, SolidWorks, Autodesk Inventor y Unigraphics NX [Erazo-Arteaga, 2022].

Los diseños generados a través de esta tecnología cuentan con relaciones geométricas, como tangencias y paralelismos, junto con restricciones que aseguran la coherencia del modelo cuando se ajustan los parámetros. Los autores [Erazo-Arteaga, 2022] y [Suárez*, et al.*, 2019] mencionan tres principios fundamentales de este enfoque:

- **Parámetros:** Son variables que definen la geometría del modelo, tales como dimensiones, ángulos, restricciones, fórmulas, propiedades de materiales. Modificar estos parámetros ajusta automáticamente la forma y el comportamiento del diseño.
- **Relaciones y restricciones:** Establece cómo interactúan los distintos elementos del modelo al realizarse cambios; las restricciones pueden ser geométricas (relaciones de los objetos) y por cota (valores geométricos).
- **Finalidad del diseño:** Garantiza que el modelo conserve su funcionalidad y diseño general a pesar de las modificaciones, gracias a las relaciones y restricciones definidas previamente.

2.3.2. Formatos y herramientas de CAD

La elección de un programa de CAD adecuado es un factor clave para el desarrollo sencillo y efectivo de una experiencia de AR. Con base en la experiencia propia, se elaboró la Tabla 2.4, utilizada para evaluar estos criterios.

Criterio	Descripción
Compatibilidad	Capacidad de integrar directamente los modelos 3D en las experiencias. Generalmente, esta integración no es directa, sino mediante la exportación e importación de los modelos, por lo que los formatos de archivo soportados por el SDK juegan un papel fundamental.
Capacidades de modelado 3D	Es la cantidad de detalle que puede añadirse a un modelo, incluyendo texturas, colores, complejidad, renderizados y herramientas en general para dar una apariencia realista.
Exportación y formatos de archivo	Es importante que el *software* de modelado tenga la capacidad de exportar a distintos formatos, para garantizar el uso en diferentes programas y cumplir con requisitos específicos.
Colaboración y flujo de trabajo	La colaboración permite que varios diseñadores trabajen simultáneamente en el mismo modelo desde diferentes estaciones de trabajo, facilitando la cooperación en tiempo real. Por su parte, las características de flujo de trabajo incluyen herramientas para asistir y facilitar el proceso de diseño, desde la concepción hasta la creación y documentación final.
Soporte y actualizaciones	Contar con documentación, tutoriales, asistencia en línea y una comunidad de usuarios facilita y acelera el desarrollo de modelos, ayudando a solucionar errores de forma rápida y crear modelos con las mejores prácticas. Las actualizaciones deben ser constantes para mejorar la funcionalidad, la compatibilidad y corregir errores.

Tabla 2.4 Criterios para la selección de herramientas de modelado 3D.

Una de las características más importantes es la exportación y los formatos de archivo; estos varían en su nivel de compatibilidad, eficiencia y capacidad de realismo. Es crucial que el formato escogido permita que el *digital overlay* se muestre de forma fluida, sin retardos perceptibles, y con un nivel de detalle y calidad suficiente para crear experiencias relevantes; ambas características deben valorarse y ponderarse para encontrar el formato que ofrezca el mejor equilibrio entre rendimiento y calidad gráfica del modelo. Los formatos más utilizados para esto se muestran en la Tabla 2.5.

Formato de archivo	Características principales	Casos de uso comunes	Compatibilidad
GLTF/GLB	- Tamaño de archivo compacto. - Soporte de texturas y animaciones. - Renderizado basado en físicas (PBR).	Aplicaciones AR en *web* y dispositivos móviles.	Amplia, especialmente en plataformas de AR.
USDZ	- Formato empaquetado. - Optimizado para iOS. - Soporta escenas complejas.	Aplicaciones AR en iOS (ARKit, Quick Look).	Altamente compatible con el ecosistema iOS.
FBX	- Rico en propiedades de animación y materiales. - Soporta animaciones esqueléticas.	Aplicaciones AR que requieren modelos detallados.	Compatible con la mayoría de programas CAD.
OBJ	- Sencillo y ampliamente compatible. - Almacena principalmente datos de geometría.	Experiencias AR básicas sin necesidad de texturas complejas.	Amplia, aunque limitada en características avanzadas.
STEP/IGES	- Alta precisión para modelos CAD. - Convertido para su uso en AR.	Aplicaciones AR industriales y de ingeniería.	Usado principalmente tras conversión a otros formatos.

Tabla 2.5 Comparación de formatos de archivo utilizados en experiencias de AR.

Además, se elaboró la Tabla 2.6, en la cual se presenta una comparación detallada de diversos programas de diseño paramétrico con base en los criterios del listado anterior. La información se obtuvo de las páginas oficiales de los fabricantes y de sitios *web* especializados en diseño CAD [PTC, 2024b, Nxrev, 2024, SourceForge, 2024, Formlabs, 2023, PTC, 2024d].

Software	Compatibilidad AR	Capacidades de Modelado 3D	Exportación y Formatos de Archivo	Colaboración y Flujo de Trabajo
FreeCAD	No compatible nativamente con ARKit, ARCore o Vuforia Studio.	Modelado 3D básico a intermedio, adecuado para proyectos de código abierto.	Admite exportación a formatos como IGES, STL y OBJ, pero con limitaciones en la calidad para AR.	Limitado, sin herramientas avanzadas de colaboración.
Creo	No compatible con ARKit o ARCore. Puede integrarse con Vuforia para experiencias de AR.	Potente en modelado 3D avanzado, ideal para industrias complejas.	Soporte para formatos como STEP, IGES, y otros utilizados en AR.	Buena colaboración con PTC Windchill.
SketchUp	Compatible con ARKit y ARCore a través de plugins y extensiones, no con Vuforia Studio.	Fácil de usar para modelado 3D simple y moderado.	Exportación a formatos como OBJ y DAE, con plugins para AR.	Buena colaboración con herramientas de Google y Trimble.
Fusion 360	Compatible con ARKit y ARCore mediante aplicaciones de terceros; limitada compatibilidad con Vuforia Studio.	Capacidades sólidas para modelado 3D detallado.	Soporta exportación a formatos como STL y OBJ, con buena integración para AR.	Excelente para colaboración en la nube con herramientas integradas.
CATIA	No es compatible nativamente con ARKit, ARCore o Vuforia Studio.	Modelado 3D de alta gama, adecuado para aeronáutica y automotriz.	Admite múltiples formatos de exportación, aunque su uso para AR es limitado.	Potente en colaboración para grandes equipos.
Inventor	Compatible de forma limitada con ARKit y ARCore; se puede integrar con Vuforia Studio.	Excelentes capacidades de modelado 3D para ingeniería.	Exporta a formatos comunes como STL, OBJ, IGES usados en AR.	Integra con otros productos de Autodesk para colaboración.
OpenSCAD	No compatible con ARKit, ARCore o Vuforia Studio.	Enfocado en modelado 3D programático, menos adecuado para AR.	Limitado a exportación en STL y otros formatos básicos.	Sin herramientas de colaboración avanzadas.
Onshape	Integración con Apple Vision Pro (basado en ARKit) y compatible con Vuforia Studio.	Modelado 3D sólido con herramientas modernas en la nube.	Exportación sencilla a formatos AR compatibles.	Excepcional colaboración en tiempo real.
Rhinoceros 3D	Compatible con ARKit y ARCore mediante plugins; no nativamente con Vuforia Studio.	Fuerte en modelado 3D y NURBS para diseño detallado.	Soporta una amplia variedad de formatos, ideal para AR.	Ofrece herramientas de colaboración limitadas.
Siemens NX	No compatible nativamente con ARKit, ARCore o Vuforia Studio.	Altamente avanzado para modelado 3D complejo y producción.	Soporta múltiples formatos para exportación, pero no optimizado para AR.	Colaboración fuerte en entornos industriales.
Solidworks	Compatible con Vuforia Studio; limitado con ARKit y ARCore.	Modelado 3D robusto para ingeniería y diseño de productos.	Soporta exportación a formatos como STL y otros compatibles con AR.	Excelente para colaboración con soluciones integradas de Dassault Systèmes.
TurboCAD	No compatible nativamente con ARKit, ARCore o Vuforia Studio.	Capacidades intermedias para modelado 3D, menos optimizado para AR.	Exportación limitada a formatos básicos como STL.	Colaboración limitada, más enfocado en usuarios individuales.

Tabla 2.6 Comparación las prestaciones para AR de algunos programas de diseño paramétrico.

2.4. Vuforia Studio

Entre los SDK de AR presentados en los antecedentes, se destacó Vuforia Studio como el más adecuado para crear experiencias alineadas con los objetivos de la presente investigación. En esta sección se analizará este *software*, comenzando con una visión general para luego explorar sus

características específicas para el ámbito industrial, sus capacidades de integración en la Industria 4.0 y las herramientas que ofrece para desarrollar experiencias complejas, inmersivas y atractivas.

PTC es la empresa creadora del Vuforia SDK, un *kit* de desarrollo utilizado por varias de sus plataformas, incluyendo Vuforia Engine, Vuforia Chalk, Vuforia Expert Capture y Vuforia Studio. Este SDK se destaca por sus funcionalidades avanzadas de visión por computadora, que permiten el reconocimiento de una amplia variedad de objetos, tanto bidimensionales como tridimensionales. Además, ofrece desarrollo multiplataforma y cuenta con un sólido soporte técnico tanto de la empresa creadora como de la comunidad en blogs y foros. A continuación se presentan algunas de sus características más importantes obtenidas de sus sitios *web* oficiales ([Vuforia Engine Team, 2024], [PTC, 2024c]):

- **Reconocimiento de imágenes y objetos:** Permite el reconocimiento y seguimiento de imágenes planas, objetos tridimensionales, múltiples imágenes en una sola `vista`, imágenes contenidas en cilindros, códigos de barras estándar y VuMarks, los cuales son marcadores personalizados. Además, soporta el seguimiento espacial basado en el entorno capturado por la cámara, así como el seguimiento de superficies planas.
- **Integración:** Se integra estrechamente con motores de desarrollo como Unity, un motor de videojuegos ampliamente utilizado, así como con los *kits* de desarrollo de AR para iOS (ARKit) y Android (ARCore), lo que facilita la creación de experiencias.
- **Multiplataforma:** Compatible con Android, iOS y algunos *headsets* de AR como Microsoft HoloLens mediante plataformas como UWP y Magic Leap OS.
- **Flexibilidad:** Se complementa con una variedad de herramientas que enriquecen y facilitan la creación de experiencias, como *Area Target*, *Model Target Generator, Vuforia Creator, Creo Parametric, Creo Illustrate, ThingWorx* y el portal de desarrolladores de Vuforia Engine. Además, es compatible con entornos de desarrollo como Unity, iOS y Android.

2.4.1. Características de Vuforia Studio

Dentro del SDK de Vuforia se encuentra Vuforia Studio, una herramienta ejecutable desde un navegador *web* para la creación de contenido de AR, diseñada para ser utilizada sin necesidad de conocimientos avanzados de programación. Esta plataforma se diferencia de Vuforia Engine en su enfoque simplificado, pero conserva las características y ventajas del *kit* de desarrollo original. Incluye funcionalidades específicas que la hacen ideal para desarrollar experiencias rápidas y sencillas en el ámbito industrial [Vuforia Engine Team, 2024]; las más importantes, según [Integral Innovation Experts, 2024] son:

- **Interfaz intuitiva:** Proporciona una interfaz de usuario intuitiva basada en arrastrar y soltar (*drag-and-drop*), lo que facilita la creación de contenido de AR.
- **Integración con IoT:** Permite integración con la plataforma IoT de PTC: ThingWorx, para crear experiencias de AR que interactúan con datos en tiempo real.
- **Métodos de seguimiento:** Facilita la creación de aplicaciones que utilizan modelos CAD, permitiendo el seguimiento a través de la cámara del dispositivo mediante reconocimiento del objeto real (*Model Target*), superficies planas (*Spatial Target*), imágenes 2D (*Image Target*), datos de geolocalización (*Area Target*) y códigos gráficos nativos de Vuforia conectados con IoT (ThingMarks).

- **Publicación y distribución:** Simplifica el proceso de publicación y distribución de aplicaciones de AR, permitiendo a los usuarios compartir y acceder a las experiencias fácilmente mediante la aplicación Vuforia View, compatible con los sistemas operativos Android, iOS y Windows Holographic.

2.4.2. Entorno de desarrollo de Vuforia Studio

En esta sección se enlistan los componentes clave del ambiente de desarrollo de PTC; entre ellos se encuentran cuatro elementos principales que, al integrarse, permiten crear experiencias de AR completas e interconectadas:

1. **Vuforia Studio:** Herramienta basada en navegador para diseñar y publicar experiencias de AR.
2. **Vuforia View:** Aplicación para acceder y visualizar experiencias de AR en dispositivos móviles. Para encontrarlas y ejecutarlas, es necesario escanear un ThingMark o códigos QR provenientes de la librería de experiencias.
3. **Experience Service:** Servidor que almacena y distribuye experiencias de AR publicadas; cada Experience Service tiene una *URL* única asignada.
4. **ThingWorx:** Plataforma utilizada para autenticar usuarios de Vuforia. Con una licencia de ThingWorx, es posible integrar datos de IoT en las experiencias de Vuforia Studio.

Para visualizar experiencias en **Vuforia View**, es necesario que éstas se encuentren publicadas en el **Experience Service**. La publicación requiere credenciales de inicio de sesión y la *URL* correspondiente al Experience Service de la organización; ambos elementos se obtienen a través de una licencia de pago. Este sistema permite gestionar y controlar el acceso, para que solo los usuarios autorizados puedan acceder al contenido.

2.4.3. Proceso de creación de experiencias de AR

Para entender el desarrollo de una aplicación de AR en Vuforia Studio, es necesario conocer el funcionamiento del entorno de desarrollo y sus conceptos clave, así como su *framework*. Finalmente, es importante preparar los recursos externos, como diseños CAD, y, dependiendo del caso de uso, aprender a utilizar plataformas adicionales como ThingWorx y Creo Illustrate.

Funcionalidades y conceptos de Vuforia Studio

Esta herramienta, que se ejecuta de forma nativa en el navegador *web* (*web native*), ofrece una interfaz visual que permite combinar elementos 2D y 3D para crear *digital overlays*. También integra una función de vista previa que se activa al presionar el botón `Vista previa`, lo que abre una ventana emergente en el navegador en la cual se simula la ejecución de la experiencia en un dispositivo móvil. Los cambios realizados en el proyecto se reflejan automáticamente en la vista previa, lo que facilita la verificación y validación de los ajustes de manera flexible, rápida y eficiente.

El contenido creado se organiza y presenta en `vistas`; este es un concepto interno de la herramienta, el cual se refiere a una pantalla o escena que se define para mostrar los elementos visuales deseados por el autor. Las `vistas` permiten organizar y estructurar la forma en que se muestra la información

virtual (*digital overlays*), que puede incluir modelos 3D, *widgets*, animaciones y acciones programadas según la lógica definida en el archivo `.js` de la `vista` correspondiente.

Es importante diferenciar un *digital overlay* de una `vista`; el primero es un concepto intrínseco a cualquier experiencia de AR, el segundo es un concepto interno de Vuforia Studio, que se refiere a la forma de organizar y definir tanto el contenido digital como la lógica de las interacciones.

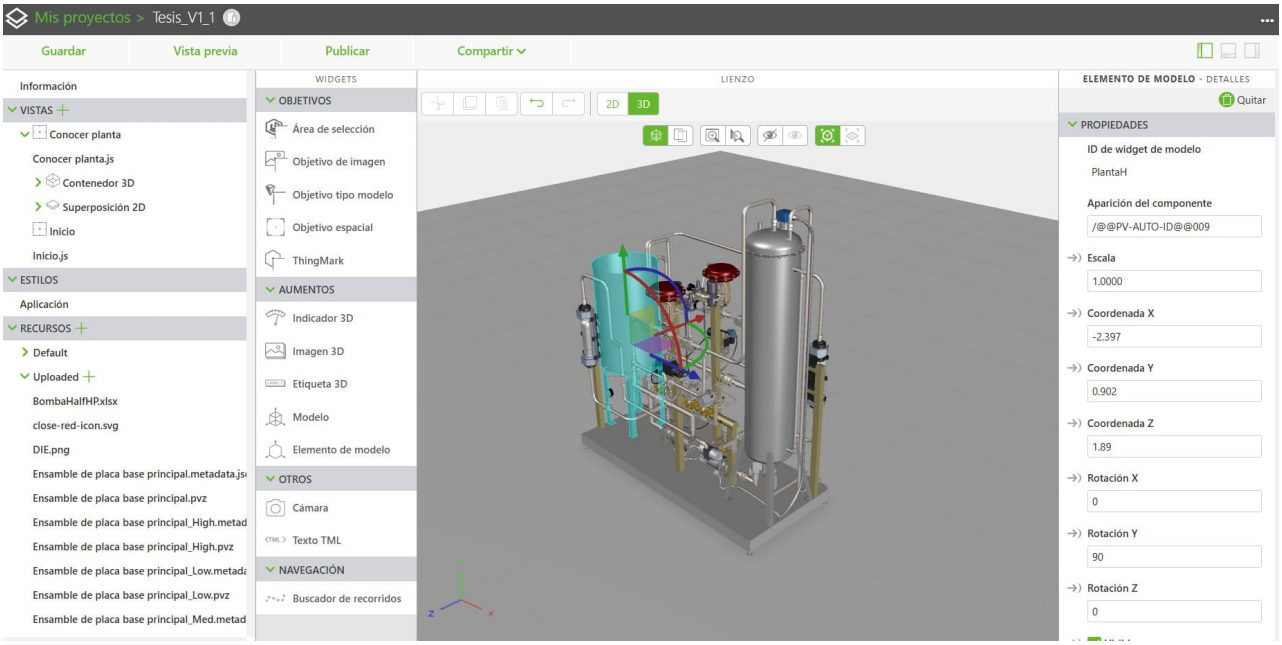


Figura 2.5 Interfaz de usuario de Vuforia Studio. [5]

Como ya se mencionó, Vuforia Studio utiliza el modelo *drag-and-drop* para que el autor diseñe las experiencias; los elementos que se pueden arrastrar dentro de una `vista` se denominan *widgets*; algunos ejemplos de estos se muestran en la Tabla 2.7. Estos elementos tienen propiedades y eventos asociados y pueden ser arrastrados tanto en el lienzo 2D como en el 3D; al ejecutar la experiencia, ambos lienzos se superponen, creando un *digital overlay* que combina elementos de ambas dimensiones.

2D		
Contendedores	**Entrada**	**Otros**
-Tarjeta	-Botón	-Audio
-Pie de página	-Casilla	-Imagen
-Diseño de cuadricula	-Selección	-Cámara
-Cabecera	-Botón para alternar	-Fichero
-Panel	-Área de texto	-Visualización de valor

3D			
Objetivos	**Aumentos**	**Otros**	**Navegacion**
-Área de selcción -Image Target -Model Target -Spatial Target -ThingMark	-Indicador 3D -Imagen 3D -Etiqueta 3D -Model -Model Item	-Cámara -Texto TML	-Buscador de recorridos

Tabla 2.7 Ejemplos de *widgets* 2D y 3D en Vuforia Studio.

[5] Imagen de autoría propia.

Las propiedades de los *widgets* se utilizan para configurar algunos aspectos gráficos como posiciones, orientaciones, visibilidad y contenido, con el fin de mostrar la información adecuada en un formato legible en el momento oportuno. Estas propiedades se pueden modificar mediante la UI del sistema, como se muestra en el lado derecho de la Figura 2.5, o a través de código en lenguaje JavaScript, en el archivo con terminación `.js` correspondiente a la `vista` que se esté trabajando, como se indica en la sección de `vistas` en el lado izquierdo de la misma imagen.

Por su parte, los eventos enlazan las acciones del usuario con comportamientos dentro de la experiencia, haciéndola interactiva en tiempo real. Un ejemplo de evento asociado a un *widget* es el de `pulsar`: cuando se presiona un botón, este puede ejecutar una acción como la navegación hacia otra `vista`, mostrando un *digital overlay* distinto. Al igual que las propiedades, las acciones pueden definirse tanto en la UI, enlazando directamente el evento al objeto destino, como mediante código JavaScript.

Vuforia proporciona una lista de eventos y propiedades generales asociadas a todos los *widgets*, accesibles mediante JavaScript. Además, ofrece propiedades y eventos específicos para cada *widget*, los cuales pueden consultarse en la siguiente fuente: [PTC, 2024a].

Otra capacidad importante para el proceso de creación es la de importar diversos recursos al proyecto, como imágenes personalizadas, archivos de audio, documentos y modelos CAD, los cuales pueden ser desplegados en las `vistas`. Esta funcionalidad permite crear experiencias inmersivas, relevantes y visualmente atractivas. Por ejemplo, la incorporación de imágenes personalizadas contribuye a establecer una identidad gráfica definida, mientras que los modelos tridimensionales realistas mejoran el nivel de inmersión de las experiencias.

Métodos de seguimiento

Cada experiencia utiliza uno o varios métodos de seguimiento, lo cual determina la manera en que será visualizada por los usuarios. Para ilustrar visualmente esto, se elaboró el diagrama de la Figura 2.6. Según la *web* oficial [PTC, 2024a], los métodos de seguimiento que este entorno de desarrollo permite son:

- ***Model Target*:** Es un *widget* utilizado para reconocer un objeto físico comparándolo con su representación 3D (modelo CAD); esto se realiza mediante visión por computadora. Alternativamente, el reconocimiento puede basarse en una vista inicial, definida por una silueta 2D creada por el desarrollador a partir del modelo 3D. En este caso, el usuario debe alinear la vista inicial con la perspectiva observada a través de la cámara del objeto real correspondiente para ejecutar la experiencia.

- ***ThingMarks*:** Son marcadores únicos identificables reconocidos universalmente por Vuforia View, similares a los códigos QR. Al ser escaneados, muestran la experiencia de AR asociada. Este tipo de seguimiento es capaz de vincular un objeto físico con datos de IoT, lo que requiere una licencia de ThingWorx.

- ***Spatial Target*:** Es un *widget* que permite colocar una experiencia de AR en cualquier superficie plana, sin necesidad de una *ThingMark*, imagen o modelo. Este tipo de seguimiento permite modificar la escala, rotar y cambiar la posición del modelo mediante gestos con los dedos.

- ***Image Target*:** Es un *widget* que realiza el seguimiento, despliegue y anclaje de la experiencia de AR basándose en la ubicación de cualquier imagen 2D definida por el desarrollador. Este

método interactúa con el entorno, ya que, al cambiar la orientación o posición de la imagen, la experiencia se desplaza con ella; asimismo, la escala del modelo 3D varía en función del tamaño físico de la imagen.

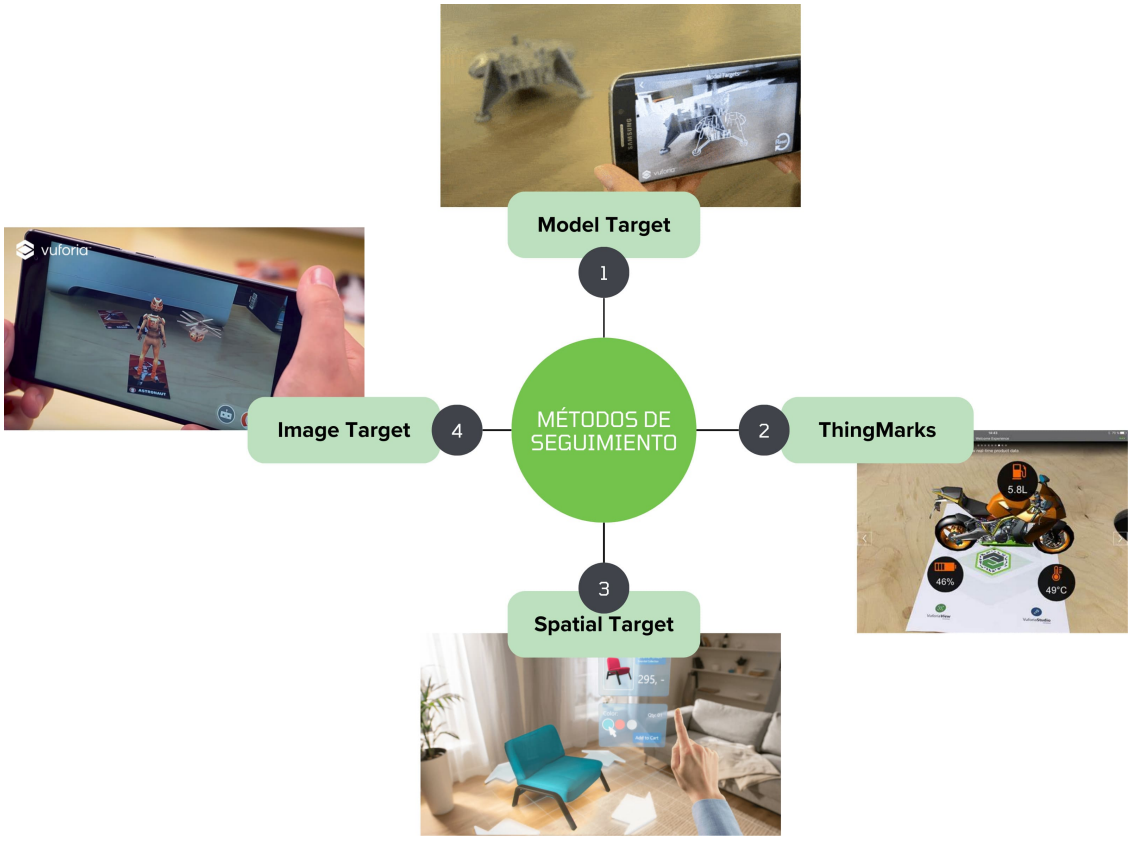


Figura 2.6 Métodos de seguimiento disponibles en Vuforia Studio. [6]

Framework de Vuforia Studio

El *framework* utilizado por Vuforia Studio guarda muchas similitudes con el diseño *web*, por lo que algunos conceptos de esta disciplina son aplicables al desarrollo de experiencias de AR [PTC, 2024a]. La estructura de estas se conforma por tres capas fundamentales, que reflejan los principios básicos del desarrollo *web*:

1. **Capa estructural:** En lugar de elementos lenguaje de marcado de hipertexto (HTML, por sus siglas en inglés), Vuforia Studio emplea *widgets* que actúan como componentes básicos dentro de una experiencia; estos son objetos interactivos que se colocan en las `vistas`. Al igual que en el diseño *web*, donde los elementos HTML son las piezas fundamentales de la estructura de una página, en Vuforia Studio los *widgets* forman la base sobre la cual se construyen las interacciones y la funcionalidad de la experiencia AR.

2. **Capa de comportamiento:** En esta capa, JavaScript se utiliza para definir la lógica y las interacciones de los *widgets*. Al igual que en el desarrollo *web*, donde JavaScript controla el

[6] Figura de autoría propia.

comportamiento de los elementos HTML, en Vuforia Studio este lenguaje dicta lo que sucede cuando un usuario interactúa con un *widget*.

3. **Capa de presentación:** CSS se emplea para controlar la apariencia visual de la experiencia AR de manera similar a su uso en el diseño *web*. Esta capa define cómo se ven los *widgets* en los *digital overlays*, especificando aspectos como colores, bordes y tipos de letra. Por ejemplo, un botón puede tener bordes redondeados y texto en color verde, todo definido por las reglas de estilo en CSS que el usuario escribe en la hoja de estilos que se puede observar debajo de la sección `Vistas`, del lado izquierdo de la Figura 2.5.

Esta estructura en capas no solo hace que el desarrollo de experiencias AR en Vuforia Studio sea intuitivo para los desarrolladores *web*, sino que también proporciona una separación clara entre la estructura, el comportamiento y la presentación, lo que facilita la creación, el mantenimiento y la personalización de experiencias. Además, el uso de JavaScript extiende las funcionalidades de los *widgets*, permitiendo incluir animaciones, secuencias y comportamientos complejos, lo que resulta en experiencias más dinámicas e interactivas [PTC, 2024a].

Herramientas complementarias

Vuforia Studio se puede complementar con Thingworx y Creo Illustrate para crear experiencias aún más relevantes para el área industrial, especialmente con enfoque en Industria 4.0. Thingworx ofrece una plataforma potente para visualizar datos de IoT; esto permite que la experiencia se conecte directamente con las máquinas y, potencialmente, con algún sistema de administración del ciclo de vida del producto (PLM, por sus siglas en inglés).

Por su parte, Creo Illustrate es un *software* que permite crear ilustraciones técnicas tanto en 2D como en 3D, así como secuencias y animaciones muy realistas y visualmente atractivas. Además, esta herramienta extiende las capacidades de integración de las experiencias creadas en Vuforia Studio, pues permite incluir modelos CAD diseñados en Creo Parametric directamente en el proyecto.

2.5. Conceptos de programación clave

2.5.1. JavaScript

Para ampliar las funcionalidades mediante código en Vuforia Studio, es necesario revisar algunos conceptos básicos relacionados con la creación de animaciones y la definición de comportamientos en las experiencias utilizando el lenguaje de desarrollo de esta plataforma: **JavaScript**.

Este es un lenguaje interpretado, es decir, el código fuente se traduce y ejecuta línea por línea durante la ejecución del programa [Mozilla Developer Network (MDN), 2024b]. Puede ser usado para crear animaciones y definir comportamientos en las experiencias de AR.

Para los fines de este trabajo, no es necesario profundizar en todos los aspectos técnicos del lenguaje; basta con familiarizarse con los tipos de variables y estructuras de control, las cuales se muestran en la Tabla 2.8, elaborada con la información proporcionada por [Mozilla Developer Network (MDN), 2024b].

Categoría	Tipo de dato/ Esctructura	Descripción	Ejemplo
Tipos de variables	Number	Número enteros y flotantes	`var pi = 3.14;` `var count = 0;`
	Bool	Verdadero y falso	`var activarBomba = true;` `var activarBomba = false;`
	String	Cadena de caracteres	`var ident = "Bomba1.5HP";`

	Object	Colección de datos y funciones	`var bomba = {`
			`modelo: "4IME0150A",`
			`RPM: 3454 };`

	Array	Colección ordenada de valores	`var listaObjetos = ['Bomba1', 'Bomba2'];`
	Null	Ausencia de valor	`var traslacionFlechas = null;`

Condicionales if	if	Ejecuta código si la condición es verdadera	`if (condición){`
			`//código`
			`}`

	if - else	Ejecuta un código si la condición es verdadera u otro si es falsa	`if (condición){`
			`//código1`
			`} else {`
			`//código2`
			`}`

	if - else if - else	Evalúa múltiples condiciones	`if (condicion1) {`
			`//código1`
			`} else if (condicion2) {`
			`// código2`
			`} else {`
			`// código3`
			`}`

	condiciones if anidadas	Condicionales if dentro de otros if	`if (condicion1) {`
			`if (condicion2) {`
			`//código`
			`}`
			`}`

Ciclos for	for	Bucle con número de iteraciones especificado	`for (inicializacion; condicion; incremento) {`
			`//código`
			`}`

	break	Detiene el ciclo antes de completar todas las iteraciones	`for (var i = 0; i <10; i++) {`
			`if (i == 5) { break;}`
			`}`

Funciones	var nombreF = function (varA, varB){ //código}	Segmento de código que realiza una tarea específica	`var activarBomba: function(bool) {`
			`estadoBomba = true;`
			`console.log('La bomba está encendida');`
			`}`

Objetos	var nombreObjeto = { //ListaObjetos}	Estructura de datos con propiedades y métodos	`var coordenadasYOriginales = {`
			`'modeloTapa': -0.202,`
			`'modeloBomba': -0.369,`
			`'Tuerca1': -0.259,`
			`'Tuerca2': -0.259,`
			`'Screw1': -0.228,`
			`'Screw4': -0.228 };`

Tabla 2.8 Tipos de variables y estructuras de control en JavaScript.

Otra herramienta de JavaScript útil para el desarrollo en Vuforia Studio es la capacidad de manipular datos en *arrays* usando métodos de estos mismos. Según [Mozilla Developer Network (MDN), 2024b], algunos son:

- `.includes(value)`: Determina si un valor específico está presente en el *array*. Devuelve `true` si el valor existe, y `false` en caso contrario.
- `.push(element)`: Añade uno o más elementos al final del *array*.
- `.splice(startIndex, deleteCount, item1, item2, ...)`: Elimina uno o más elementos del *array* a partir de la posición especificada por el índice `startIndex`. También permite añadir nuevos elementos en esa posición.

- `.indexOf(value)`: Devuelve el primer índice en el que se encuentra el elemento especificado. Si el elemento no está presente, devuelve `-1`.

En el entorno de desarrollo de Vuforia Studio, es útil crear **objetos** en JavaScript con diversos propósitos. Estos pueden entenderse como colecciones de datos y funcionalidades, compuestas por atributos (variables) y métodos (funciones). En el paradigma de programación orientada a objetos, los objetos se crean a partir de una plantilla denominada clase, es decir, son instancias de dicha clase [Mozilla Developer Network (MDN), 2024b].

Por ejemplo, en el Código 2.1, se define de forma explícita un objeto llamado `bomba`, el cual contiene atributos (`info, ident`) y métodos (`activarBomba, desactivarBomba`). En la función `activarBomba`, la palabra reservada `this` hace referencia a las propiedades del objeto actual. Al ejecutar este método, el mensaje mostrado en la consola será 'La Bomba1.5HP modelo 4IME0150A está encendida'.

```
var bomba = {                                    // Se define un objeto llamado 'bomba'
  info: {                                        // La propiedad 'info' es otro objeto que almacena informacion
      modelo: "4IME0150A",                       // El modelo de la bomba, almacenado como una cadena de texto
      RPM: 3454                                  // Variable de tipo entero, almacena las RPM de la bomba
  },                                             // Cierre del subobjeto 'info'
  ident: 'Bomba1.5HP',                           // La propiedad 'ident' almacena un identificador para la bomba
  activarBomba: function(bool) {                 // El metodo 'activarBomba' es una funcion que enciende la bomba
    this.estadoBomba = true;                     // Establece la propiedad 'estadoBomba' a 'true'
    console.log('La ${this.ident} modelo ${this.info.modelo} esta encendida');  // Imprime un mensaje en consola
  },                                             // Cierre del metodo 'activarBomba'
  desactivarBomba: function(bool) {              // El metodo 'desactivarBomba' es una funcion que apaga la bomba
    this.estadoBomba = false;                    // Establece la propiedad 'estadoBomba' a 'false'
    console.log('La '+this['ident']+' modelo '+ this['info']['modelo']+' esta apagada');   // Mensaje en consola
  }                                              // Cierre del metodo 'desactivarBomba'
};                                               // Cierre del objeto 'bomba'
```

Código 2.1 Ejemplo de la definición de un objeto.

Para acceder a los métodos o atributos de un objeto, se utilizan de forma indistinta dos notaciones comunes: la **notación de punto** y la **notación de corchetes**. En el Código 2.2, se accede al atributo `ident` con la notación de punto para asignarle la cadena 'Bomba1.5HP'. En la siguiente línea, se emplea la notación de corchetes para modificar el atributo `RPM`, perteneciente al subobjeto `info`, que a su vez es un atributo del objeto `bomba`, y se le asigna el valor 3454:

```
bomba.ident = 'Bomba1.5HP';                      // Asignacion de una cadena a la propiedad ident
person["info"]["RPM"] = 3454;                    // Asignacion de un entero a la propiedad RPM del subobjeto info
```

Código 2.2 Accediendo y modificando las propiedades de un objeto.

Además de lo visto anteriormente, JavaScript permite trabajar archivos con formato JSON, por sus siglas en inglés *JavaScript Object Notation*, que es un formato de texto estándar utilizado para representar y transmitir datos estructurados. En JavaScript, la manipulación de este tipo de datos se realiza mediante métodos como `JSON.parse()`, que transforma una cadena JSON en un objeto, y `JSON.stringify()`, que realiza el proceso inverso, convirtiendo un objeto de JavaScript en una cadena de texto JSON [Mozilla Developer Network (MDN), 2024b].

En Vuforia Studio, este formato es útil para obtener una copia de la información de un *widget* sin asignación de referencia. Al convertir un *widget* (que es realmente un objeto de JavaScript) en un archivo JSON y luego volver a convertirlo a un objeto de JavaScript, es posible clonar toda la información del *widget* y realizar modificaciones en él sin perder sus datos originales. En el Código 2.3 se muestra esta funcionalidad.

```
var ModelItem = {                                    // Objeto original llamado ModelItem
  coordenadas: {x:0, y:0, z:0},                      // Sub-objeto de coordenadas, atributo de ModelItem
};                                                   // Cierre de ModelItem

$scope.obtenerParametros = function() {              // Funcion para clonar parametros de ModelItem
  cordsModelItem = JSON.parse(JSON.stringify(ModelItem)); //Clona las propiedades de ModelItem en cordsModelItem
};

$scope.modificarParametros = function() {            // Funcion para modificar parametros de corsModelItem
cordsModelItem.coordenadas.x = 10;                   // Las modificaciones no afectan a ModelItem
};
```

Código 2.3 JSON para copiar un objeto sin asignación de referencia.

2.5.2. AngularJS

AngularJS es un *framework* que sigue el patrón de diseño MVC (Modelo-Vista-Controlador), por lo que organiza el código en tres componentes: el modelo (datos), la vista (interfaz de usuario) y el controlador (lógica que conecta ambos), manteniendo una conexión constante entre las partes (*data binding*). AngularJS permite conectar la lógica de la aplicación con el diseño visual, asegurando que cualquier cambio en los datos se refleje automáticamente en la interfaz de usuario [Parada, 2021].

Vuforia Studio incluye un editor de código en JavaScript que se utiliza en las `vistas` de la aplicación y se ejecuta dentro del contexto de un controlador específico de AngularJS. Esto significa que el código escrito se integra en el controlador, permitiendo manipular directamente los datos y comportamientos de la aplicación a través de objetos nativos como `$scope` [PTC, 2024a]. En este entorno de desarrollo, el controlador de AngularJS es responsable de manejar la lógica de la aplicación y coordinar los datos entre el modelo y la vista, permitiendo la interacción dinámica y la manipulación de datos en las experiencias de AR.

Al escribir código JavaScript en el archivo `.js` de una `vista`, la primera línea de código indica los servicios y objetos nativos del controlador que pueden ser utilizados por el desarrollador para agregar funcionalidades en la experiencia; estos se muestran en el Código 2.4.

```
// $scope, $element, $attrs, $injector, $sce, $timeout, $http, $ionicPopup, and $ionicPopover services are
   available
```

Código 2.4 Objetos de AngularJS disponibles en Vuforia Studio.

En el contexto de este desarrollo, el objeto de AngularJS más utilizado es **`$scope`**, este es crucial para la comunicación entre el controlador y la vista, pues permite que los datos y funciones definidos en el controlador estén disponibles en la vista y viceversa. Además, `$scope` facilita la actualización y reflejo de cambios en la vista en tiempo real [AngularJS, 2022].

El objeto `$scope` tiene dos usos principales: primero, definir funciones que realizan modificaciones o actualizaciones en la vista, y segundo, acceder a las propiedades y parámetros de los *widgets*. En el Código 2.5 se muestra la definición de una función llamada `quitarZoomModelo` dentro del objeto `$scope`, así como la modificación de propiedades de los *widgets* presentes en la `vista`.

```
$scope.quitarZoomModelo = function(){              // Se define la funcion quitarZoomModelo en el objeto $scope

  $scope.view.wdg['PlantaH']['scale'] = 0.1;       // Se establece la propiedad scale del widget 'PlantaH' en 0.1
  $scope.view.wdg['PlantaH']['x'] = 0;             // Se establece la propiedad x del widget 'PlantaH' en 0
  $scope.view.wdg['PlantaH']['z'] = 0;             // Se establece la propiedad z del widget 'PlantaH' en 0

  $scope.$apply();                                 // Se ejecuta el metodo $apply() para actualizar la vista
};                                                 // Cierre de la funcion quitarZoomModelo
```

Código 2.5 Usos del *$scope* en Vuforia Studio.

La función `quitarZoomModelo` se define en `$scope` porque las acciones que realiza modifican la representación que muestra la `vista`; en este caso, cambia las coordenadas `'x'` y `'z'` del *widget* llamado `'PlantaH'`.

La sintaxis general para modificar parámetros es: `$scope.view.wdg['ID de Studio']['propiedad']`, donde el `ID de Studio` corresponde al nombre definido por el desarrollador para cada *widget*, tratándose de una clave única que el programa reconoce.

Respecto a los métodos nativos del objeto `$scope`, esta investigación se centrará únicamente en la función **$apply()**, ya que es la única usada en el desarrollo. La función evalúa una expresión en el contexto de AngularJS y actualiza tanto el modelo como la vista [AngularJS, 2022]. `$apply()` es fundamental para realizar animaciones en el entorno de Vuforia Studio, ya que permite que los cambios en las propiedades de los *widgets* se reflejen en la `vista`. En el Código 2.5, las modificaciones en la escala y coordenadas se hacen visibles en la `vista` gracias a la actualización realizada por esta función en la línea 7.

Entre los servicios nativos incluidos en el controlador de AngularJS, dos son especialmente relevantes para el desarrollo de este trabajo. El primero es **`$interval`**, que permite ejecutar una función repetidamente a intervalos de tiempo específicos [AngularJS, 2022], es decir, ejecuta una función cada cierto número de milisegundos determinado. Este servicio puede devolver una "promesa" que se notifica en cada iteración del intervalo y se resuelve después de un número específico de iteraciones, o bien, puede ejecutarse indefinidamente si no se define un número de iteraciones.

Otra funcionalidad de este servicio es cancelar un intervalo llamando al método `$interval.cancel()`. En el Código 2.6, la función `secuenciaAnimacion` se ejecuta cada 10 milisegundos y evalúa el cumplimiento de una promesa; si esta se resuelve, asigna el resultado de `iniciarAnimacion` a `animacionAgua`; por el contrario, si esta arroja un error, se llama a `$interval.cancel()` para detener la ejecución de `secuenciaAnimacion`.

```
1 var secuenciaAnimacion = $interval(function () {          // Variable de referencia para la funcion $interval
2     promesa.then((resolve) => {                           // Si la promesa se resuelve, se ejecuta el metodo then
3     animAgua = iniciarAnimacion();                        // Guarda el resultado de iniciarAnimacion en animAgua
4     }).catch((error) => {                                 // Si ocurre un error en la promesa, se ejecuta catch
5       $interval.cancel(secuenciaAnimacion);               // En caso de error, se ejecuta $interval.cancel
6     });
7 }, 10);                                                   // La funcion secuenciaAnimacion se ejecuta cada 10 ms
```

Código 2.6 Uso del *$interval* en Vuforia Studio.

Por su parte, el servicio **`$timeout`** permite ejecutar una función después de un retraso especificado en milisegundos. Al igual que `$interval`, este servicio devuelve una 'promesa' que se resuelve una vez que el retraso ha pasado y la función se ha ejecutado; además, puede ser cancelado llamando a `$timeout.cancel()` [AngularJS, 2022]. En el entorno de Vuforia Studio, este servicio puede usarse para realizar animaciones o detectar actualizaciones en los datos después de un tiempo específico. Por ejemplo, en el Código 2.7, se define una función llamada `switchview` que cambia la `vista` después de un retraso de 2 segundos.

```
1 $scope.app.switchView = function(destination) {           // Ejecuta una funcion tras un tiempo especificado
2   $timeout(function() {                                   // Retrasa la ejecucion de la funcion
3     twx.app.fn.navigate(destination);                     // Cambia de vista de la experiencia
4   }, 2000);                                               // Tiempo especificado, 2000 milisegundos = 2 segundos
5 };
```

Código 2.7 Uso del *$timeout* en Vuforia Studio.

El concepto de **promesa**, mencionado en los servicios anteriores, se refiere a objetos que son usados para conocer el estado de una tarea asíncrona, es decir, una tarea que puede ejecutarse de

manera independiente sin esperar a que una tarea previa termine. Las promesas permiten una gestión efectiva de las operaciones que no se completan de inmediato [AngularJS, 2022]. Los posibles estados que adoptan son:

1. **Pendiente (`pending`):** La promesa sigue en operación; es decir, no ha sido completada ni rechazada.
2. **Cumplida (`fulfilled`):** La promesa se cumple; además, devuelve un valor.
3. **Rechazada (`rejected`):** La promesa no se cumple; además, devuelve un error.

Además, las promesas constan de algunos métodos:

- `.then (resolve)`: Ejecuta una función cuando la promesa se cumple.
- `.catch (reject)`: Ejecuta una función cuando la promesa es rechazada.
- `.all (<Array de promesas>)`: Ejecuta una función cuando todas las promesas, que se encuentran en el *array* que se proporciona como argumento, se cumplen. Si se da el caso de que tan solo una de las promesas es rechazada, automáticamente se va al caso `.catch()`.

2.5.3. CSS

CSS es un lenguaje de programación utilizado para especificar el estilo y los aspectos visuales de las páginas *web*. Surge en 1996 para superar las limitaciones de HTML en cuanto a estilos gráficos. Fue presentado como una alternativa a la inclusión de estilos directamente en el código de la página mediante etiquetas como `<font>` [Alvarez, 2020]. Este lenguaje permite definir con precisión distancias y posiciones de los elementos gráficos utilizando una variedad de unidades, como `cm, px, %, in`, entre otras [Alvarez, 2020]; además, es útil para modificar propiedades como visibilidad, colores, márgenes, opacidad y muchas otras.

Los selectores permiten aplicar estilos gráficos a elementos específicos dentro de una interfaz. Existen varios tipos de selectores, como los de tipo o etiqueta, de clase y de *ID* [Tinoco y Solís, 2014]. En el contexto de Vuforia Studio, se emplean selectores de clase, los cuales utilizan la sintaxis presentada en el Código 2.8 y permiten definir las propiedades gráficas de los *widgets* que tienen dicha clase asignada.

```
1 .NombreDefinido{                          // Define una clase con nombre NombreDefinido
2   propiedad: valor;                       // Atributos graficos que seran aplicados a los objetos
3   propiedad: valor;
4 }
```

Código 2.8 Selector de clase en CSS.

Muchas propiedades de CSS utilizan valores de unidades de medida para definir aspectos como tamaños de imágenes, fondos, textos, así como posiciones, márgenes y dimensiones, entre otros. Los valores de longitud se dividen en dos categorías: absolutos y relativos. Los valores absolutos, como se muestra en la Tabla 2.9, no dependen de ningún otro elemento y se consideran siempre del mismo tamaño. Los valores relativos, por otro lado, dependen del tamaño de un contenedor o del contexto gráfico, como se detalla en la Tabla 2.10. En general, las unidades más utilizadas en Vuforia Studio son `px`, `vh` y `vw`.

Unidad	Nombre	Equivale a
cm	Centímetros	1cm = 96px/2,54
mm	Milímetros	1mm = 1/10 de 1cm
Q	Cuartos de milímetros	1Q = 1/40 de 1cm
in	Pulgadas	1in = 2,54cm = 96px
pc	Picas	1pc = 1/6 de 1in
pt	Puntos	1pt = 1/72 de 1in
px	Píxeles	1px = 1/96 de 1in

Tabla 2.9 Unidades de longitud absoluta en CSS. [Mozilla Developer Network (MDN), 2024a].

Unidad	Relativa a
ex	Altura x de la fuente del elemento.
ch	La medida de avance (ancho) del glifo "0"
rem	Tamaño de la letra del elemento raíz.
lh	Altura de la línea del elemento.
vw	1 % del ancho de la ventana gráfica.
vh	1 % de la altura de la ventana gráfica.
vmin	1 % de la dimensión mínima de la ventana gráfica.
vmax	1 % de la dimensión máxima de la ventana gráfica.

Tabla 2.10 Unidades de longitud relativa en CSS. [Mozilla Developer Network (MDN), 2024a].

A continuación se clasifican algunas de las propiedades que pueden ser modificadas en Vuforia Studio usando CSS.

Fuente de texto

Las propiedades mencionadas por [Tinoco y Solís, 2014] en este apartado que se pueden modificar son:

1. **`font-family:`** Es una agrupación de tipos de letra con características similares; existen cinco familias genéricas de fuentes: `Serif`, `Sans-serif`, `Cursive`, `Monospace`, `Fantasy`, se utiliza de esta forma para que el navegador pueda encontrar un tipo de letra en una misma familia; de forma que si no encuentra la primera tipografía, utilice la siguiente, por ejemplo: *Arial, Helvetica, sans-serif*
2. **`font style:`** Definir un estilo de letra normal, inclinada o cursiva.
3. **`font-variant:`** Controla variaciones para las fuentes; algunas son: `small-caps`: Convierte texto a letras pequeñas mayúsculas. `All-small-caps`: Convierte todo a pequeñas mayúsculas. `Petite-caps`: Pequeñas mayúsculas más pequeñas que `small-caps`.
4. **`font-weight:`** Controla la intensidad o grosor de las fuentes; algunos valores son: `normal, bold, bolder, lighter, 100`, etc.
5. **`font-size:`** Controla el tamaño de las fuentes; los valores son: `xx-small`, `x-small`, `small`, `medium`, `large`, `x-large`, `xx-large`, `large`, `smaller`, etc.
6. **`font:`** Es la forma comprimida, en la cual se pueden definir todos los valores de las fuentes; los más usados son: `font-family`, `font-style`, `font-variant`, `font-weight`, `font-size`.

Formato de texto

Son las propiedades utilizadas para controlar la apariencia y disposición del texto [Tinoco y Solís, 2014]; incluye el estilo, la alineación, espaciado y transformación del texto:

1. **`text-indent:`** Permite añadir sangría en la primera línea en los párrafos de texto.
2. **`text-align:`** Alinea el texto a la derecha, izquierda o centro del elemento que lo contiene; los valores son: `left`, `right`, `center`, `justify`.

3. **`text-decoration`:** Permite otorgar diferentes atributos a los textos, como subrayar, tachar, remarcar o parpadear un texto; los valores principales son: `underline`, `overline`, `line-thorugh`, `blink`.

4. **`letter-spacing` y `word-spacing`:** Se utilizan para definir la distancia entre caracteres y entre palabras en un texto.

5. **`text-transform`:** Cambia un texto original por un texto definido por las siguientes propiedades: mayúscula (valor de `uppercase`), minúsculas (valor de `lowercase`) o con la primera letra de cada palabra en mayúscula (`capitalize`).

6. **`text-overflow`**: Controla cómo se muestra el texto que se desborda de su contenedor cuando el texto es más largo que el espacio disponible.

Colores y fondo

Controlan todas las propiedades relacionadas con colores e imágenes para el fondo, texto y todos los elementos visuales que puedan adoptar un color [Tinoco y Solís, 2014]. Consta de las siguientes propiedades:

- **`color`**: Define el color de texto del documento; se puede especificar con nombres en inglés (`black`, `silver`, `white`, etc.) o mediante el formato `RGB (valor rojo, valor verde, valor azul)`, con valores entre 0 y 255. Ejemplo: el azul es `RGB(0, 0, 255)`.

- **`background-color`**: Determina el color de fondo de los elementos. En Vuforia Studio, puede aplicarse a `widgets` de texto, botones, paneles, menús emergentes (*pop-ups*) y a varios elementos en la `vista` 2D y algunos en la `vista` 3D.

- **`background-image`**: Inserta una imagen de fondo. Sus casos de aplicación son similares a los de `background-color`.

- **`background-repeat`**: Controla si la imagen de fondo se repite en uno o ambos ejes. Valores posibles: `repeat`, `no-repeat`, `repeat-x`, `repeat-y`.

- **`background-attachment`**: Define si el fondo se desplaza con el contenido o permanece fijo respecto al contenedor. Valores: `scroll`, `fixed`.

- **`background-position`**: Establece la posición inicial del fondo dentro de un elemento. Valores: `top`, `bottom`, `center` para vertical; `left`, `center`, `right` para horizontal; o coordenadas como 50 % 50 %.

- **`background`**: Propiedad abreviada que combina varios aspectos del fondo (imagen, color, repetición, posición, etc.). Ejemplo: `background: url(image.jpg) no-repeat center;`.

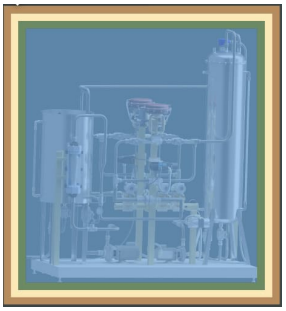
Cabe aclarar que, en Vuforia Studio, cualquier imagen que se desee utilizar como fondo para un *widget* debe ser añadida previamente al proyecto. Esto se realiza mediante el botón **`Añadir recurso`**, ubicado en la sección de `Recursos` en el lado izquierdo de la interfaz (Figura 2.5). Una vez añadida, la *URL* de la imagen tiene la siguiente estructura: `#$resources/Uploaded/NombreImagen.svg`, donde `NombreImagen` debe ser reemplazado por el nombre asignado a la imagen por el autor, y debe conservar el tipo de archivo de la imagen.

Modelo de caja

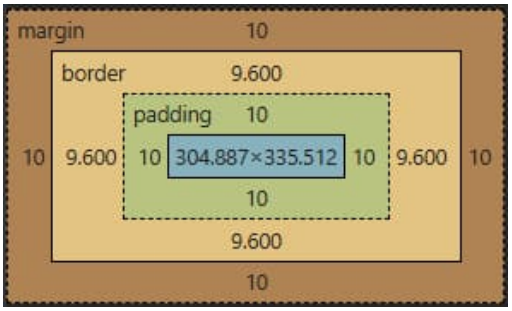
El modelo de caja en CSS describe la estructura y el espacio ocupado por los *widgets*. Cada uno de estos tiene un contenedor rectangular compuesto por 4 áreas, las cuales utilizan valores en unidades de longitud: margen, borde, relleno y contenido [Tinoco y Solís, 2014]. El siguiente listado describe dichas áreas:

- **Contenido:** La parte central de la caja, donde se coloca el texto, imágenes o cualquier contenido visual.
- **Relleno (`padding`):** El espacio entre el contenido y el borde interior de la caja. No afecta al color del borde y es completamente transparente, pero amplía el tamaño del área total de la caja; las propiedades son: `padding-top`, `padding-right`, `padding-bottom`, `padding-left` y `padding`.
- **Borde (`border`):** Una línea alrededor del `padding` y el contenido. El tamaño, color y estilo del borde se pueden modificar con las propiedades de CSS, como `border-width`, `border-style` y `border-color`.
- **Margen (`margin`):** El espacio exterior entre el borde del elemento y los elementos adyacentes. No afecta el color de la caja, pero define el espacio entre cajas contiguas; las propiedades asociadas son: `margin-top`, `margin-right`, `margin-bottom`, `margin-left` y `margin`.

Un ejemplo práctico de cómo se representan estas áreas se muestra en la Figura 2.7a, donde cada área está resaltada con un color distinto. La Figura 2.7b proporciona una representación simplificada, mostrando los nombres de las áreas y los valores correspondientes en `px`.



(a) Ejemplo del modelo de caja.



(b) Versión simplificada del modelo de caja.

Figura 2.7 Modelo de caja de CSS. [7]

Diseño responsivo con *Media Queries*

Los *media queries* permiten aplicar estilos CSS en función de las características del dispositivo, como el tamaño de la pantalla, definido por el ancho (`width`) y el alto (`height`) de la ventana en píxeles, así como su orientación: vertical (`portrait`) u horizontal (`landscape`). La combinación de estas propiedades permite crear interfaces que mantienen una apariencia visual consistente en distintas orientaciones y resoluciones [Mozilla Developer Network (MDN), 2024a]. Por ejemplo, en el Código 2.9 se ajusta el tamaño del fondo de un botón teniendo en cuenta estas características.

[7] Figuras de autoría propia.

```
//------Orientacion horizontal----------------//
@media(orientation: landscape) {                    // Aplica cuando la pantalla esta en modo horizontal
  /* Pantallas Estrechas (Menos de 580 px de ancho) */
  @media (max-width: 580px) {                       // Aplica cuando el ancho maximo de la pantalla es 580px
    .b_close{                                       // Clase que sera asignada a los objetos deseados
      background-size: 3.5vw 3.5vw;                 // Tamano del fondo ajustado a pantallas pequenas usando vw
    }                                               // Cierre de la clase .b_close
  }                                                 // Cierre del Media querie para el ancho de pantalla
  /* Pantallas Muy Amplias (Mas de 1024 px de ancho) */
  @media (min-width: 1025px) {                      // Aplica cuando el ancho maximo de la pantalla es 1025px
    .b_close{                                       // Clase que sera asignada a los objetos deseados
      background-size: 2.5vw 2.5vw;                 // Tamano del fondo mas pequeno para pantallas grandes en vw
    }                                               // Cierre de la clase .b_close
  }                                                 // Cierre del Media querie para el ancho de pantalla
}                                                   // Cierre del Media querie para la orientacion
```

Código 2.9 Utilización de *Media queries*.

Posiciones

La propiedad `position` define el método de posicionamiento aplicado a un elemento, determinando cómo se ubica en el documento o dentro de su contenedor. Esta propiedad se complementa con `top`, `right`, `bottom` y `left`, que en conjunto determinan la ubicación final del elemento en la `vista` [Mozilla Developer Network (MDN), 2024a]. Las interacciones entre estas propiedades, así como sus valores, se detallan en la Tabla 2.11.

Valor de position	Descripción	Comportamiento	Propiedades asociadas
`static`	Valor predeterminado. El elemento se posiciona según el flujo normal del documento.	No se puede ajustar su posición con top, right, bottom, o left.	`Ninguna`
`relative`	El elemento se posiciona en relación con su posición original.	Se puede mover con `top, right, bottom, left,` pero su espacio original sigue reservado.	`top, right, bottom, left`
`absolute`	El elemento se posiciona respecto a su contenedor más cercano con position distinta de static.	Elimina el elemento del flujo normal del documento, permitiendo su posición específica.	`top, right, bottom, left`
`fixed`	El elemento se posiciona en relación a la ventana gráfica.	Permanece fijo en la misma posición al hacer scroll. Sale del flujo normal.	`top, right, bottom, left`
`sticky`	Comienza como relative y se convierte en fixed cuando se cumple una condición de desplazamiento.	El elemento “se pega” en una posición mientras el usuario se desplaza en la vista.	`top, right, bottom, left`

Tabla 2.11 Tipos de posicionamiento en CSS.

Propiedades adicionales

Algunas propiedades adicionales que no se categorizan en las secciones anteriores, pero que son usadas para el desarrollo en Vuforia Studio, y que son mencionadas en [Mozilla Developer Network (MDN), 2024a] se muestran en el siguiente listado:

1. **Opacidad (`opacity`)**: Controla el nivel de transparencia de un elemento; se expresa en valores de 0 (completamente transparente) a 1 (completamente opaco). Por ejemplo, si un botón tiene

`opacity: 0.5`, será semitransparente, por lo que será posible ver tanto el contenido del fondo como al mismo *widget*.

2. **Transformaciones (`transform`)**: Permite modificar el espacio de coordenadas del modelo, de forma que los elementos pueden ser trasladados, rotados, escalados o sesgados de acuerdo a los valores establecidos. Algunos ejemplos de transformaciones son: `rotate(deg)`, `transform: scale(sx, sy)`, `translate(tx, ty)`, entre otras. Una propiedad relacionada es `transform-origin`, la cual define el punto desde el cual se aplican las transformaciones.

3. **Cursor (`cursor`)**: Cambia el aspecto del cursor al pasar sobre el elemento indicado. Un valor común es `pointer` para indicar que el elemento es interactivo.

2.6. Sistema de tanques atmosféricos interconectados

Como se indicó en los objetivos, la experiencia de AR se implementará en una planta de manejo de líquidos, la cual cumple con las características de una instalación industrial [Mobley, 2001]. Esta instalación se utiliza en asignaturas de la División de Ingeniería Eléctrica (DIE) de la Facultad de Ingeniería (FI) de la UNAM, donde los estudiantes desarrollan sistemas de control y supervisión, interactuando regularmente con ella. Se espera que la incorporación de una experiencia de AR en estas actividades facilite la comprensión del sistema, contribuyendo a validar la utilidad de la herramienta en el ámbito operativo de sistemas industriales y al cumplimiento de los objetivos de esta investigación. Por ello, resulta pertinente describir dicha instalación.

Figura 2.8 Renderizado de un modelo CAD de la instalación industrial. [8]

2.6.1. Descripción del sistema

El sistema de tanques atmosféricos interconectados es una instalación industrial para el manejo de líquidos, ubicada en el Laboratorio de Automatización de la D.I.E. Este sistema consta de dos tanques de almacenamiento de líquidos a presión atmosférica: uno con techo abierto y otro con techo cerrado, interconectados mediante tuberías galvanizadas de 1 y 3/4 de pulgada de diámetro. El sistema de bombeo incluye dos bombas centrífugas accionadas por motores eléctricos trifásicos. Además, a lo largo de las tuberías se encuentran diversos instrumentos de medición, como transductores de flujo y nivel, medidores analógicos de presión y flujo y válvulas de paso manuales [Santamaría, 2024]. En la Figura 2.8 se presenta un renderizado del modelo CAD que ilustra dicha instalación.

En adición a lo anterior, se presenta la Tabla 2.12, realizada con la información proporcionada por [Santamaría, 2024], la cual incluye datos técnicos e imágenes representativas de los instrumentos.

2.6.2. Funcionamiento de la instalación de manejo de líquidos

La interconexión de los tanques, mostrada en la Figura 2.8, está conformada por dos válvulas de corte Swagelok accionadas de forma independiente por una válvula MAC series 900; además, las dos válvulas de control Orion serie 9000 son accionadas cada una por un transductor Fisher 646, los cuales controlan el nivel de apertura de las válvulas. A lo largo de las tuberías hay ocho válvulas de accionamiento manual, junto con diversos instrumentos de medición: dos rotámetros Fischer&Porter, cuatro transmisores de presión diferencial SMAR LD301 y un sensor de nivel Warrick Controls 3E2C instalado en el tanque cubierto [Santamaría, 2024].

La integración de la instrumentación con un sistema de control automático, compuesto por un PLC y otros equipos eléctricos como interruptores termomagnéticos y variadores de frecuencia, permite controlar y monitorear el nivel de cada tanque de forma independiente. A su vez, habilita el desarrollo de sistemas de supervisión basados en los datos proporcionados por la instrumentación y transmitidos al PLC, lo que permite la creación de pantallas interfaz humano-máquina (HMI, por sus siglas en inglés) y sistemas SCADA. Esto tiene como objetivo permitir que ingenieros integradores programen lógica de control y supervisión, funcionando como una herramienta práctica para aplicar conocimientos profesionales con instalaciones industriales y *software* profesional.

[8] Obtenido de: [Santamaría, 2024].

Instrumento	Descripción	Especificaciones técnicas		Imagen alusiva
		Parámetro	Magnitud	
Válvula de control Orion serie 9000	Válvula acoplada a un actuador neumático modelo PA35	No. de serie:	196143	
		Tamaño de conexión:	1.0 in	
		Rango de señal neumática :	3-15 PSIG	
		Flujo máximo(Cv) :	8.3	
		Presión máxima de trabajo :	60 PSIG	
Válvula de corte Swagelok SS-65TS16	Válvula de bola tipo ON/OFF, unida a un actuador neumático modelo Swagelok serie 133	Presión/Temp. max.	2200 PSIG/100°F	
		Tamaño de conexión:	1.0 in	
		Presión max. del actuador	200 PSIG/37°C	
		Presión de actuación	25-150 PSIG/37°C	
		Flujo máximo (CV)	40	
Válvula neumática MAC series 900	Válvula de 4 vías que controla la presión suministrada al actuador de la válvula de corte	Tamaño de conexión	1/4 IN NPTF	
		Número de puertos:	4	
		Presión de operación	25 - 150 PSI	
		Modelo	912B-PM-501JM	
		Flujo máximo (CV)	1.4	
Placa de orificio Emerson	Placa concéntrica sin acabado, usada para medir caudal, instalda mediante dos bridas.	Modelo:	Rosemount 1495	
		Espesor:	0.125 in	
		Material:	Acero inoxidable 316SS	
		Diametro interno:	0.735 in	
		Diseñada para:	Tubería de 2-24 in	
Rotámetro Fischer&Porter	Medidor de caudal de área variable. Medición observada mediante una escala y un flotador.	Rango de medición:	0-30 gal/min	
		Precisión de lectura:	±2 %	
		Presión max de trabajo:	150psig / 200°F.	
		Diametro de conexión:	4 in	
		Material de la carcasa:	Acero inoxidable 316SS	
Tanque de techo abierto	Tanque atmosférico con tomas de agua y vidrio de nivel para observación visual.	Tomas de agua:	Inferior	
		Vidrio de nivel:	Lateral izquierdo	
		Conexión de llenado:	Parte superior trasera	
		Toma de presión:	Parte frontal inferior	
		Material de construcción:	Acero inoxidable 316SS	
Tanque atmosférico de techo cerrado	Tanque de acero con tomas de agua, de drenaje y transf. conexiones de llenado y trasnferencia.	Material de construcción:	Acero inoxidable 316SS	
		Tomas/conexiones de agua:	Parte inferior/superior	
		Sensor de nivel conductivo:	Warrick controls 3E2C	
		Transmisor de presión:	Moore Serie 340D	
		Vidrio de nivel:	Lateral derecho	
Manómetro análogo	Manómetro para medir la presión manométrica, i.e. relativa a la atmósfera.	Tipo de presión:	Manométrica	
		Aplicación:	Calibrar transductor	
		Compatible con:	Transductor Fisher 646	
		Uso en:	Tuberías de conexión	
		Lectura:	Carátula análoga	
Transductor Fisher 646	Transductor neumático, convierte una señal de corriente de entrada en presión de salida.	SERIAL No.:	11691670	
		Entrada:	4 to 20 mA	
		Salida:	3 to 15 psi	
		Temp. de operación:	-40° to 160°F	
		Conexiones de presión:	1/4 NPT	
Regulador de filtro Fisher 67CFR	Regulador para mantener presión constante en sistemas neumáticos y electro-neumáticos	Presión de entrada:	250 PSIG	
		Presión de salida:	0 a 125 PSIG	
		Rango de temperatura:	-40° a 160°F	
		Conexiones de presión:	1/4 NPT	
		Filtración de partículas:	5 micrómetros	
Transmisor de presión diferencial SMAR LD301	Transmisor inteligente con control PID con en sensor capacitivo, para medir presiones de fluidos.	Tipo de sensor:	Capacitivo	
		Rango de medición:	4-20 mA	
		Función de transferencia:	Seleccionable	
		Terminal de comunicación:	HART	
		Montaje:	Abrazaderas	
Transmisor de presión diferencial Moore Serie 340D	Sensor de presión diferencial con PID, para detección y control de procesos.	Señal de salida:	4-20 mA	
		Protocolo de comunicación:	HART	
		Rango de temperatura:	0-85 °C	
		Rango de presión:	Hasta 1.25 kPa	
		Material de construcción:	Acero 316SS y aluminio	

Tabla 2.12 Instrumentación del sistema de tanques atmosféricos.

Capítulo 3

Desarrollo

3.1. Preparación, importación y uso de modelos CAD

Los modelos CAD constituyen el núcleo de los *digital overlays*, al ser el elemento visual predominante en todas las `vistas`. Un tratamiento previo adecuado, siguiendo las mejores prácticas, asegura una importación correcta y facilita el desarrollo de animaciones y secuencias en la experiencia; por ello, esta sección describe el proceso de tratamiento previo, importación y manejo de modelos tridimensionales en Vuforia Studio.

La documentación de PTC incluye un manual de buenas prácticas para crear experiencias de AR en Vuforia Studio; en este se subraya la importancia de los modelos CAD para desarrollos en entornos industriales [Pitser, 2021]. El documento proporciona recomendaciones clave para la incorporación efectiva de activos 3D, las cuales fueron seguidas para la preparación y manejo de los modelos usados en este proyecto.

Vuforia Studio, mediante convenios de colaboración, puede recibir modelos directamente desde programas como Siemens NX, JT y CATIA V5 [Pitser, 2021]. Sin embargo, en este desarrollo se importaron los modelos utilizando formatos compatibles descritos en [PTC, 2024a], específicamente el formato `.IGES`.

Este formato fue elegido debido a que, al ser importado en un proyecto, conserva aspectos visuales como texturas, colores y escalas, así como la **simplificación** de piezas, realizada con Autodesk Inventor. Este último proceso fue crucial para la creación de los *digital overlays* en 3D, ya que cada pieza simplificada en el *software* de CAD se puede asignar directamente a un *widget* de `elemento de modelo`; la importancia y uso de este principio se explican en detalle en las subsecciones 3.1.2 y 3.4.

En cuanto al rendimiento, [Pitser, 2021] recomienda que los modelos CAD cumplan con criterios específicos: bajo consumo de memoria (menos de 20 megabytes), procesamiento eficiente y un conteo de polígonos inferior a 500,000 por `vista`. Los archivos `.IGES` son ventajosos en este sentido, ya que permiten su optimización mediante el `Optimizador de ficheros CAD` de Vuforia Studio, manteniendo los ajustes realizados en el *software* de diseño 3D.

Una vez establecidas las herramientas usadas y características esperadas del modelo, se describe el proceso de preparación previo a la importación en un proyecto.

3.1.1. Revisión y corrección de los modelos 3D

El proceso de creación de la experiencia de AR partió de modelos CAD previamente desarrollados, por lo que el tratamiento previo a su importación se limitó a la optimización del modelo, la separación y simplificación de las partes para su uso individual mediante *model items* en Vuforia Studio, así como a la mejora de la calidad visual de algunos elementos. Todas estas tareas se realizaron en Inventor, por lo que los nombres de las funcionalidades mencionadas en esta sección hacen referencia a este *software*.

Los modelos usados para esta aplicación corresponden a cualquiera de los dos tipos siguientes:

1. **Piezas:** Se trata de un conjunto de operaciones geométricas y espaciales vinculadas entre sí que modelan objetos físicos específicos, como tornillos, tuercas o placas de materiales.

2. **Ensambles:** Consisten en conjuntos de piezas unidas mediante relaciones de ensamblaje, permitiendo que funcionen como una unidad claramente diferenciada.

Antes de iniciar la preparación de los modelos, es recomendable organizar todos los archivos relacionados con un sistema o máquina en una sola carpeta. Inventor facilita esta tarea mediante la creación de un proyecto.

Respecto al rendimiento, se eliminaron las piezas que no serían visibles o utilizadas durante la experiencia, suprimiéndolas directamente desde el árbol del proyecto del ensamble. Esta acción también fue útil para la opción `Capacitación` de la `vista` `Conocer planta`, pues se utilizó un ensamble auxiliar, conformado únicamente por los elementos de la planta que serían destacados mediante animaciones. Al importar este modelo en Vuforia Studio, bastó igualar las coordenadas del modelo auxiliar con las del modelo principal para alinear todos los instrumentos de la planta. Los detalles del uso de este ensamble se encuentran en la sección 3.4.4.

Para los aspectos visuales, colores y algunas texturas, se realizaron ajustes usando el panel de materiales y aspectos, ubicado en la parte superior de la interfaz de Inventor. Esto fue utilizado para aplicar color y textura a la base de metal negro del sistema de tanques, los postes que sostienen la instrumentación, los vidrios de nivel, los transmisores de presión y los rotámetros.

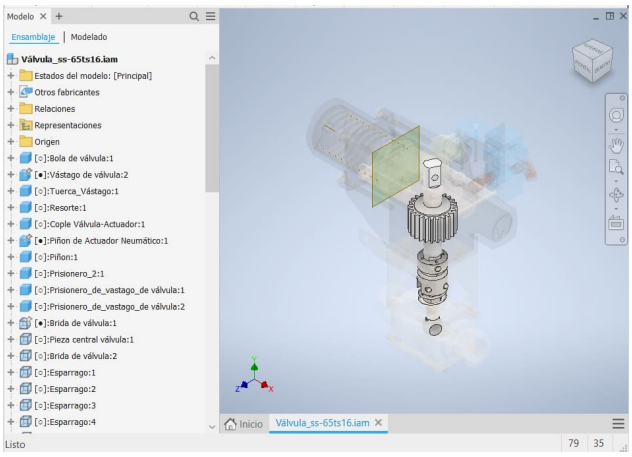
La última tarea fue la simplificación de piezas en Inventor; este proceso implica agrupar componentes de un ensamble de manera que facilite la creación de animaciones y efectos visuales. Por ejemplo, para animar la apertura de una válvula de corte, es necesario aplicar una rotación a todas las piezas involucradas en el movimiento rotacional de la válvula.

Si no se realiza el proceso de simplificación, al arrastrar un `model item` en el eje de movimiento de la válvula, se aislaría solo una de las piezas que lo componen; por lo tanto, para realizar la animación completa, se tendría que colocar un `model item` para cada una de las piezas originales (ver Figura 3.1d), lo cual dificultaría la gestión de las animaciones y elevaría el consumo de CPU de la aplicación.

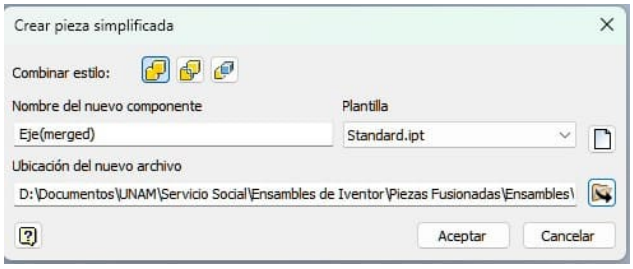
Para evitarlo, se creó una pieza simplificada, como la mostrada en la Figura 3.1d. Este proceso consistió en ocultar la visibilidad de todas las partes del ensamble ajenas a las piezas de interés; en la Figura 3.1a se destacan dichas piezas. Posteriormente, se utilizó la herramienta `Crear pieza simplificada`; al hacerlo, se configuró la pieza asignándole un nombre y cambiando la plantilla a `Standard.ipt`, asimismo, la ubicación del archivo se definió en la carpeta del proyecto previamente creado, como se muestra en la Figura 3.1b.

Después de realizar esta configuración, se rompió el vínculo con el componente base y se guardó la pieza. Luego, se insertó la pieza simplificada en el ensamble original y se utilizaron restricciones para hacer coincidir la posición de la pieza simplificada con las piezas originales (Figura 3.1c). Posteriormente,

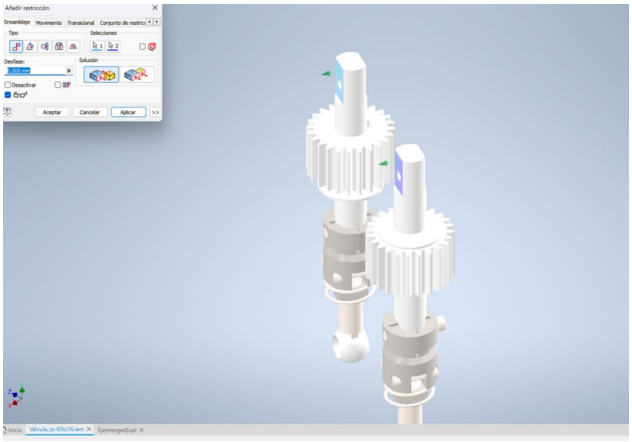
se eliminaron las piezas originales y se volvieron a hacer visibles el resto de componentes del ensamble. Como resultado, las piezas relacionadas con el movimiento rotacional de la válvula de corte se pueden aislar en Vuforia Studio usando un solo `model item`.



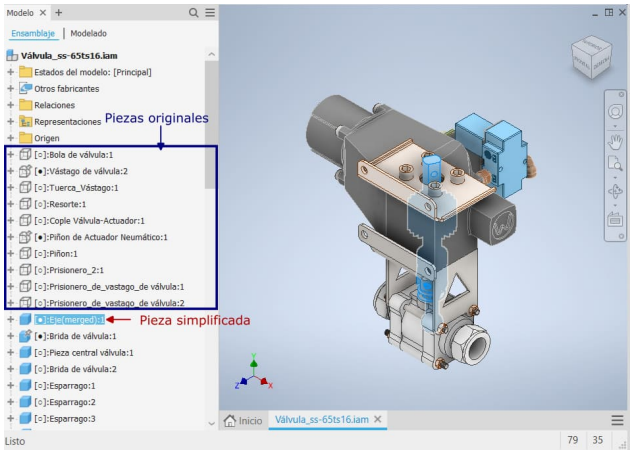
(a) Piezas elegidas para ser simplificadas.



(b) Configuración de la pieza simplificada.



(c) Uso de restricciones para alinear la pieza simplificada.



(d) Resultado de la simplificación.

Figura 3.1 Proceso de simplificación de piezas en Inventor. [1]

En el CAD de los tanques atmosféricos, el proceso de simplificación se aplicó a sensores, transductores y actuadores que no se encontraban aislados. De manera similar, se agruparon en piezas únicas todas las tuberías del circuito de transferencia y las del circuito de recirculación, permitiendo modificarlas de forma independiente (ver sección 3.3.3). Además, los instrumentos pertenecientes a cada `vista` fueron simplificados de la misma forma, considerando las necesidades de animación y resaltado en la experiencia.

[1] Figuras de autoría propia.

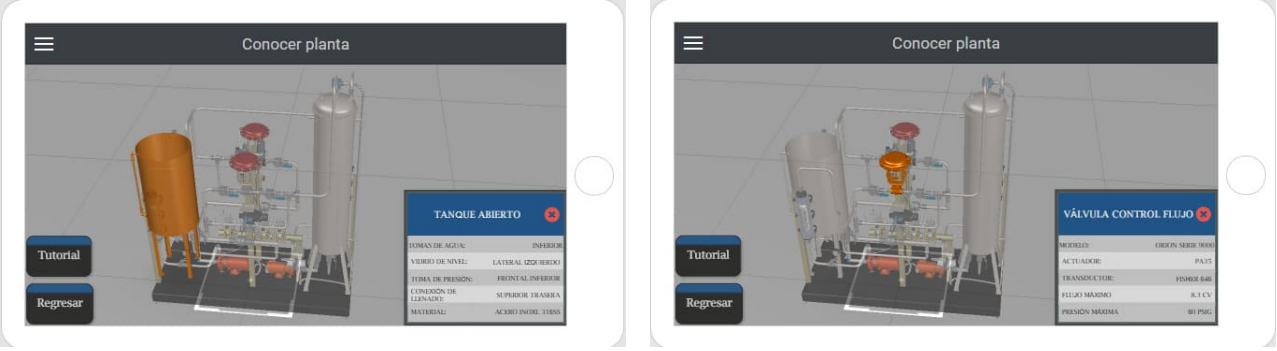
3.1.2. Importación e implementación en Vuforia Studio

Una vez que el modelo CAD cumplió con las características requeridas, se importó al proyecto en Vuforia Studio a través del panel de `Recursos`, ubicado en el lateral izquierdo de la UI del programa (ver Figura 2.5). Al seleccionar e importar el modelo, se activó la opción `Ejecutar optimizado de CAD`, lo que reduce el peso del modelo y el uso de recursos de procesamiento gráfico del dispositivo, esto sin comprometer la calidad visual. Como resultado, fue posible usar el modelo en la experiencia y el formato cambió de `.IGES` a `.PVZ`.

Tras completar este proceso, se agregó un `modelo` en el lienzo 3D, habilitando un panel de propiedades para este *widget*. En la propiedad `Recurso`, se seleccionó el ensamble del modelo CAD, haciéndolo visible. Al tener una referencia visual del aspecto del modelo, fue posible ajustar propiedades como `Escala`, `Coordenadas` y `Rotación` de forma visual y rápida, tanto en código JavaScript como directamente en la UI.

Luego, fue necesario separar las partes o piezas del sistema para construir los *digital overlays*. Se utilizó el *widget* `elemento de modelo`, arrastrando el puntero con el *clic* izquierdo sostenido desde este *widget* hasta la pieza específica en el lienzo 3D. Este proceso aísla cada equipo del sistema de tanques, permitiendo modificar sus propiedades sin afectar el modelo completo. Además, cada *model item* cuenta con eventos y propiedades independientes, lo que permite la ejecución de funciones asociadas a partes específicas; esto se observa en la Figura 3.3b.

Para ilustrar esto, en la Figura 3.3a se presenta el árbol de un proyecto 3D, donde se muestran los *model items* del sistema de tanques atmosféricos, que corresponden a los instrumentos de medición y actuadores. En los ejemplos de las Figuras 3.2a y 3.2b se utilizó el evento `pulsar` en cada instrumento para dos tareas: ejecutar animaciones de cambio de color y mostrar un panel emergente, para lo cual se crearon enlaces con el servicio `mostrar emergente` de un *widget* `pop-up`.



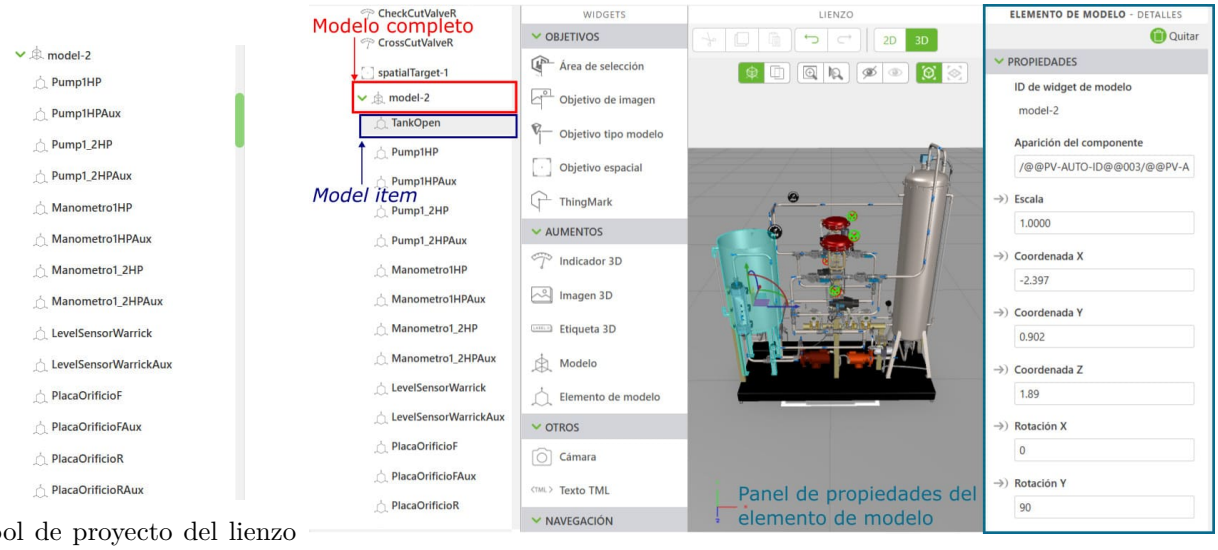
(a) Válvula de control destacada.

(b) Tanque abierto destacado.

Figura 3.2 Panel emergente con información técnica y animación de cambio de color. [2]

El resultado es un *overlay* que despliega tarjetas informativas y resalta con un cambio de color cualquier instrumento o actuador tocado por el usuario, como se muestra en las Figuras 3.2a y 3.2b. Otro ejemplo que ilustra el uso de modelos y *model items* en Vuforia Studio se encuentra en la sección 3.3.3.

[2] Figuras de autoría propia.
[3] Figuras de autoría propia.



(a) Árbol de proyecto del lienzo 3D.

(b) Propiedades de un *model item*.

Figura 3.3 Uso de modelos y *model items* en la creación de un *digital overlay*. [3]

3.2. Estilos en CSS

La capa de presentación en las experiencias de AR es crucial, ya que representa el primer punto de contacto del usuario; por ello, es esencial que sea intuitiva y atractiva. Además, mantener una coherencia gráfica también contribuye a transmitir profesionalidad y seriedad en las experiencias. En el desarrollo de la aplicación para los tanques atmosféricos interconectados, se ha prestado especial atención a la creación de *digital overlays* con estética profesional.

El trabajo en este sentido se divide en dos áreas principales: por un lado, los aspectos decorativos y, por el otro, el posicionamiento y la escala. Lo decorativo incluye atributos como colores, tipografías, niveles de opacidad, imágenes de fondo y todos los elementos visuales de los *widgets* utilizados en las `vistas`.

En cuanto al posicionamiento y escala, se garantizó que los elementos visuales estuvieran localizados estratégicamente para facilitar la interacción sin obstruir el campo de visión del usuario, y que todos los *widgets* mantuvieran proporciones y posiciones correctas en cualquier dispositivo y orientación. Este capítulo explicará cómo se implementaron estas características mediante la hoja de estilos en CSS.

3.2.1. Aspectos gráficos

Los aspectos visuales de los *widgets*, tanto en 2D como en 3D, se definieron asignándoles clases con las características gráficas deseadas. En el lienzo 3D, la calidad gráfica de los archivos CAD fue el aspecto visual más relevante, tema que ya se abordó en la sección anterior. Para los elementos tridimensionales adicionales, como etiquetas y letreros 3D, se aplicó la misma metodología utilizada para los *widgets* 2D.

En el apartado 2D, los *widgets* más utilizados incluyen: `Panel`, `Imagen`, `Etiqueta`, `Diseño de Cuadrícula (grid layout)`, `Botón`, `Elemento Emergente (pop-up)` y `Botón de Alternar`. Se emplearon propiedades como `margin` y `background-color` en la portada para crear figuras geométricas decorativas.

Las clases se definen dentro de la hoja de estilos CSS de la experiencia, la cual contiene todos los parámetros estéticos aplicables a todos los *widgets* de cualquier `vista`. Esto incluye propiedades adicionales como la tipografía (línea 1 del Código 3.1) y algunos botones con imágenes de fondo.

El código generado para este proyecto puede dividirse en las siguientes secciones, las cuales aparecen en el mismo orden dentro de la hoja de estilos CSS:

1. **Aspectos del lienzo:** Incluyen la incorporación de la fuente de texto para la experiencia y modificaciones generales para mejorar el aspecto visual, como disminuir algunos márgenes y deshabilitar algunas barras deslizantes innecesarias.

2. **Menús de operación:** Definen las propiedades de los menús emergentes ubicados a los lados y en la parte inferior de los *digital overlays*, así como los botones para navegar entre las `vistas` y pantallas.

3. **Controles de secuencia:** Agrupa todos los botones utilizados para operar las secuencias de animaciones, junto con los indicadores del paso actual (se pueden ver en las Figuras 3.12).

4. **Portada:** Configuraciones para los logotipos, textos y contenedores presentes en la portada.

5. **Ficha técnica:** Estilos y configuraciones del `pop-up` que contiene la ficha técnica desplegada al tocar la instrumentación del modelo 3D.

6. **Diseño responsivo:** Garantiza que los *widgets* se escalen y posicionen correctamente en cualquier dispositivo y orientación. Aquí se definen los intervalos para el escalamiento y posicionamiento, tanto en orientación vertical como horizontal.

Los nombres de las clases definidas para este proyecto siguen una nomenclatura, donde se indica el contenedor o la `vista` donde fue usado, seguido del tipo de *widget* o un nombre alusivo a su función en la experiencia. Algunos ejemplos se explican en el Código 3.1.

```
1 //Linea para importar la fuente al proyecto, esta debe ser agregada a los recursos en un formato permitidio
2 @font-face { font-family: 'Lucida Bright'; src: url('#{$resources}/Uploaded/LucidaBright.woff') format('woff'),}
3
4 .M1_bt_Capac{/*Parametros:Valores*/}                    // Boton en el Menu 1 para overlay de capacitacion
5 .M1_BackBut{/*Parametros:Valores*/}                     // Boton en el Menu 1 para regresar
6 .Oper-M2{/*Parametros:Valores*/}                        // Menu 2 ubicado en la vista de operacion
7 .P_title-label{/*Parametros:Valores*/}                  // Etiqueta del titulo ubicada en la portada
```

Código 3.1 Nombres definidos para las clases e importación de fuente en CSS.

Portada

La portada constituye un ejemplo que ilustra el método utilizado para la definición de estilos dentro de la aplicación. Su diseño se inspiró tanto en proyectos de muestra de Vuforia Studio como en el trabajo previo desarrollado por colaboradores de esta tecnología en la Facultad de Ingeniería. El bosquejo inicial, mostrado en la Figura 3.4, presenta la siguiente estructura básica: el logotipo de la DIE. en la parte superior, los escudos de la FI y la UNAM en el borde inferior y, en el panel central, el título de la experiencia acompañado de un renderizado del sistema de tanques.

Entre los elementos destacados se encuentra la creación de los bordes superior e inferior, generados mediante código CSS, lo que evitó el uso de *widgets* en el lienzo 2D. Para lograr cada uno de estos

[4]Figura de autoría propia.

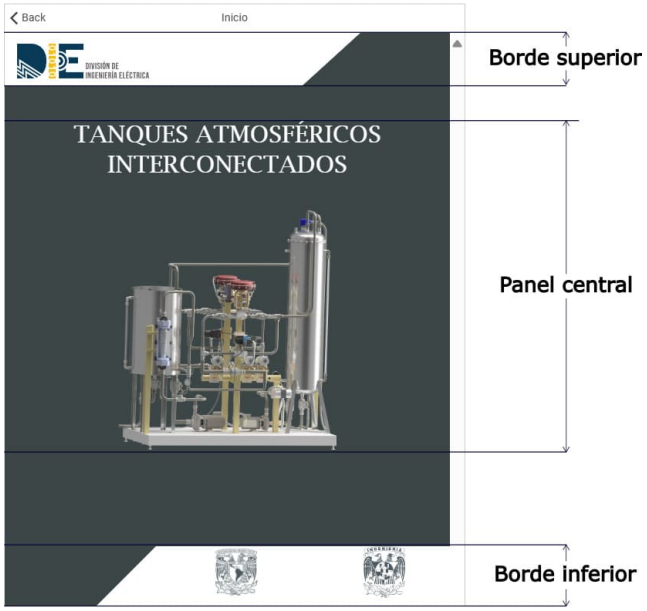


Figura 3.4 Diseño y distribución de la portada de la experiencia. [4]

bordes, se modificaron las propiedades de dos `paneles` en Vuforia Studio, como se muestra en el Código 3.2, donde se definen las clases para el borde superior (`.P_Header`) y el inferior (`.P_Footer`). Cada clase fue asignada a la primera fila de un *widget* de `diseño de cuadrícula`, colocados en los `paneles header-1` (borde superior) y `PanelCentral` (borde inferior), respectivamente.

```
.P_Header {                                    // Clase para el borde superior de la portada
  border-top: 9vh solid #fff;                  // Borde superior con ancho de 9vh, color blanco
  border-right: 10vh solid transparent;        // Borde derecho transparente para crear forma
  width: 80%;                                  // Ancho de la clase P_Header, 80% del contenedor
}

.P_Footer {                                    // Clase para el borde inferior de la portada
  border-bottom: 10vh solid #fff;              // Borde inferior con ancho de 10vh, color blanco
  border-left: 10vh solid transparent;         // Borde izquierdo transparente para forma
  width: 80%;                                  // Ancho de la clase P_Footer, 80% del contenedor
  position: fixed;                             // Fija la posicion del pie de pagina
  right: 0px;                                  // Coloca el pie de pagina en el extremo derecho
  bottom: 0px;                                 // Coloca el pie de pagina en la parte inferior
  padding: 0px;                                // Elimina el espacio interno en P_Footer
}
```

Código 3.2 Uso de propiedades de CSS para crear los bordes de la portada.

La creación de la figura mediante propiedades CSS se logró modificando los bordes y márgenes del `panel` que se genera de forma nativa al añadir una fila en un `diseño de cuadrícula`. Se definió un borde superior de 9vh de alto y 80 % del ancho de la `vista`, con un relleno sólido de color blanco en formato hexadecimal. Para crear la forma trapezoidal, se añadió un borde derecho sólido con fondo transparente, lo que genera la ilusión de una forma triangular. Gracias a que la altura predeterminada de este contenedor es cero, los únicos objetos visibles en ese panel son sus bordes.

Uso de la consola del navegador para definir estilos

Vuforia Studio permite visualizar los cambios realizados en la `vista` mediante una ventana emergente en el navegador *web*, activada a través del botón `Vista Previa` ubicado en la parte superior de la UI mostrada en la Figura 2.5. Las modificaciones pueden efectuarse ya sea desde la hoja de estilos

en CSS del proyecto, las secciones de código JavaScript de las `vistas` o directamente en el `lienzo de trabajo`. Sin embargo, los cambios solo se reflejan en la `vista previa` después de guardar las modificaciones, lo cual se puede hacer con el comando `Ctrl+S` en el teclado o mediante el botón `Guardar`, ubicado en la esquina superior izquierda.

Este método resulta poco eficiente al trabajar con estilos gráficos, ya que incluso para ver el efecto de un cambio menor, como ajustar un color o una tipografía, es necesario repetir todos los pasos descritos. Para optimizar y agilizar el diseño, se utilizó la consola de desarrollo del navegador, que permite reflejar en tiempo real los cambios realizados a una clase de CSS, siempre que esté asignada a un *widget*.

Al escoger la herramienta `Seleccionar un elemento de la página para inspeccionarlo` y posicionar el cursor sobre cualquier elemento gráfico en la pestaña `Vista Previa: Vuforia Studio`, es posible ver los parámetros gráficos del objeto seleccionado en un panel del lado derecho de la consola. De manera similar, en el panel `</> Elementos` se puede navegar por las líneas de código HTML, lo que resalta en un color diferente el elemento gráfico correspondiente a cada línea.

Una vez localizado el *widget* deseado, se probaron distintos valores para sus parámetros hasta identificar aquellos que cumplían con los requisitos de diseño. Posteriormente, dichos valores se incorporaron en la hoja de estilos CSS del proyecto, logrando que la `vista previa` reflejara el mismo aspecto observado en la consola de desarrollo, pero de manera más ágil.

Para ilustrar este proceso, se asignaron valores a los parámetros gráficos de un texto en un *widget* `etiqueta`, utilizando la clase `.P_title-label`. Los pasos seguidos se enumeran en la Figura 3.5 y se describen en el siguiente listado:

1. Se presionó la tecla `F12` para abrir la consola de desarrollo del navegador y se escogió la herramienta `Seleccionar un elemento de la página para inspeccionarlo`.
2. Se posicionó el cursor sobre el *widget* deseado, en este caso, la etiqueta.
3. El código HTML correspondiente fue resaltado automáticamente.
4. Se localizó el panel de `Estilos` en el lado derecho de la consola de desarrollo.
5. Se buscó la clase `.P_title-label` dentro del panel de `Estilos`.

Después de realizar este proceso, fue posible escribir código CSS debajo de la clase correspondiente; los cambios se reflejaron inmediatamente en la `Vista previa` sin necesidad de guardar, como se muestra en la Figura 3.6b.

Una vez determinadas las propiedades y valores adecuados para la aplicación, como se aprecia en la Figura 3.6c, se transcribió el código a la hoja de estilos del proyecto en Vuforia Studio y se guardaron los cambios; esto generó el Código 3.3.

```
.P_title-label {                    // Clase para dar estilo grafico al titulo de la experiencia en la portada
  color: white;                                          // Texto de color blanco
  font-family: 'Lucida Bright', sans-serif;              // Fuente personalizada, con reserva sans-serif
  font-size: 7vw;                                        // Tamanio de la fuente
  line-height: 5vh;                                      // Altura de linea ajustada a 5vh
  text-align: center;                                    // Texto alineado al centro
  text-transform: uppercase;                             // Convierte el texto a mayusculas
}
```

Código 3.3 Clase asignada a la etiqueta de título de la portada.

[5]Figura de autoría propia.
[6]Figura de autoría propia.

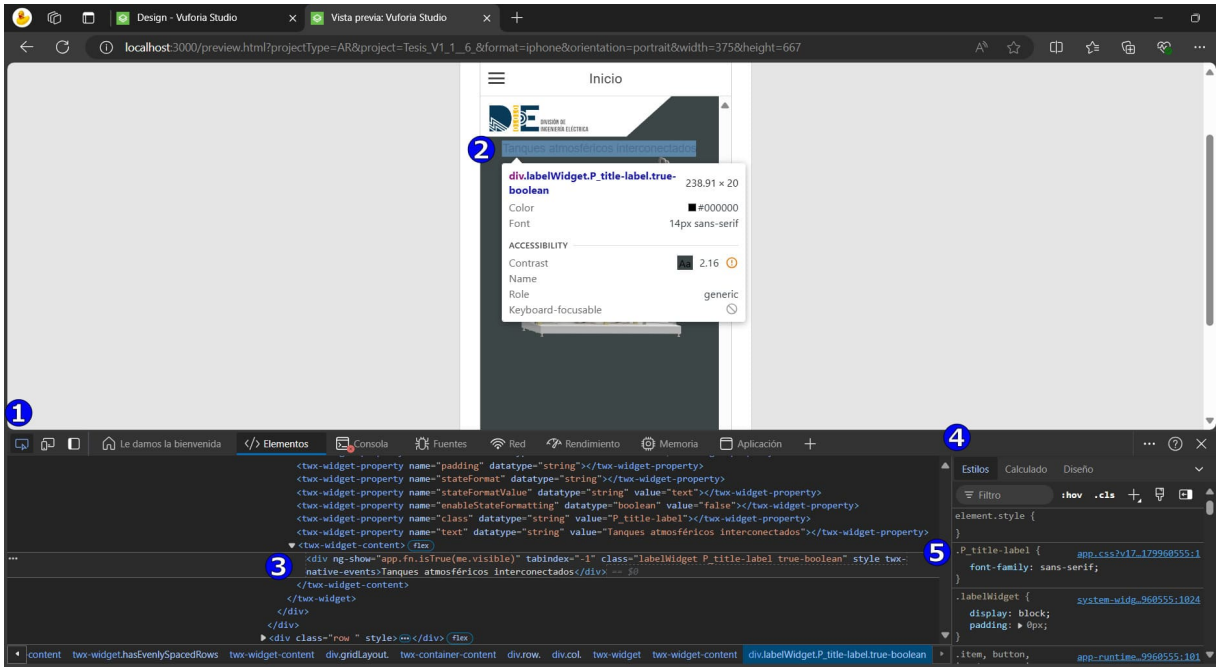
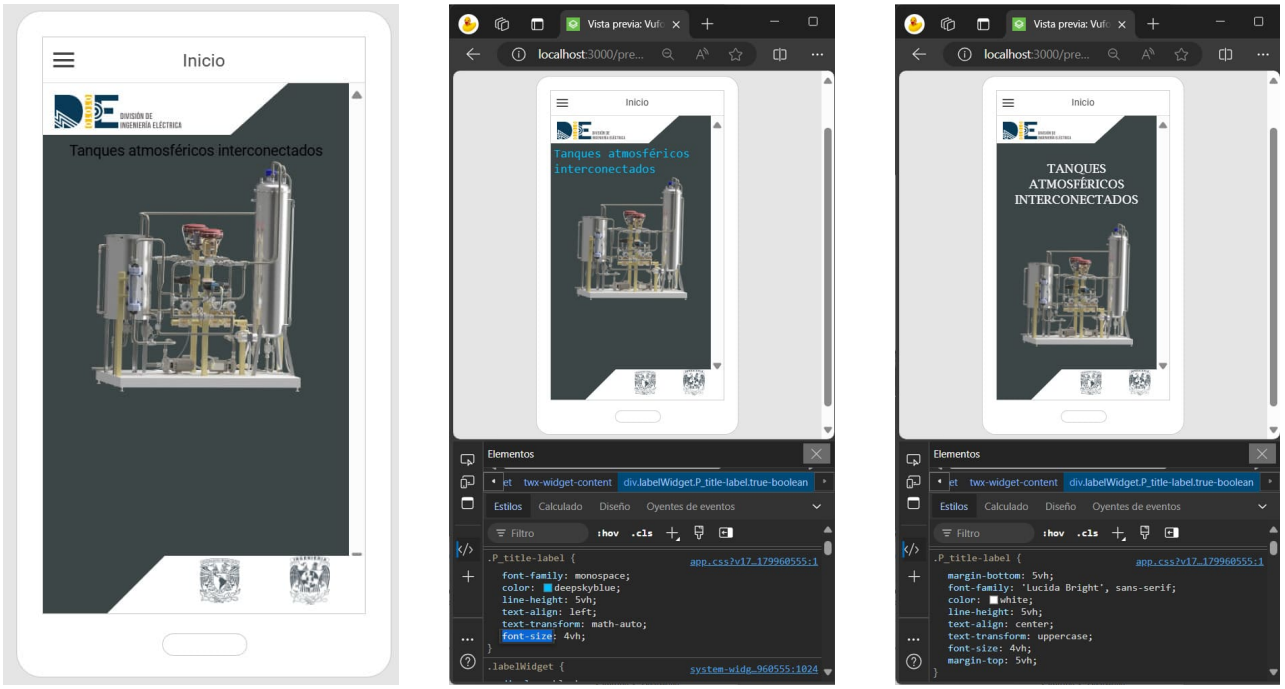


Figura 3.5 Uso de la consola de desarrollo para explorar elementos. [5]



(a) Texto sin estilo gráfico. (b) Cambio de parámetros. (c) Estilo final visto en consola.

Figura 3.6 Cambios en valores de parámetros de CSS usando la consola de desarrollo. [6]

3.2.2. Posicionamiento y escalas

Un problema recurrente al diseñar una `vista` 2D en Vuforia Studio y revisar la `vista previa` fue que la pantalla diseñada tenía un aspecto correcto en ciertos dispositivos, pero en otros presentaba problemas de posición, tamaño inadecuado o un pobre aprovechamiento del espacio. Esto sucedió

porque algunas características como la resolución, las proporciones de pantalla y el tamaño de la `vista` varían entre dispositivos.

Abordar estos problemas es crucial, ya que la experiencia será utilizada en una amplia gama de dispositivos por ingenieros de automatización. Garantizar una visualización correcta en todos ellos es esencial para evitar problemas de usabilidad y funcionalidad, como botones o imágenes con dimensiones inadecuadas, que podrían afectar negativamente la interacción del usuario.

Para resolverlo, se implementó un diseño responsivo utilizando *media queries*; este proceso consiste en ajustar los valores de las propiedades de una clase en CSS en función de las características del dispositivo en el que se visualiza la imagen. En este caso, se utilizarán dos criterios principales: la orientación del dispositivo y las dimensiones de la pantalla.

El primer criterio se implementó usando la sentencia `@media(orientation: portrait/landscape)`, dentro de la cual se incluyó el código necesario para ajustar los valores según la orientación del dispositivo, ya sea horizontal o vertical respectivamente. Para el segundo criterio, se utilizaron las propiedades de ancho y alto mínimos y máximos que proporciona la `Vista previa` en Vuforia Studio. Con estos datos, se definieron intervalos para clasificar las pantallas en las categorías: estrechas, medianas, amplias y muy amplias.

Los datos del ancho y largo de las pantallas, definidos en el entorno de Vuforia Studio, se obtuvieron a través de la consola de desarrollo del navegador en la `vista previa`; esta pestaña permite cambiar el dispositivo en el que se visualiza la experiencia. Usando la herramienta de `inspeccionar elemento`, de la misma forma que se describió en la sección anterior, fue posible obtener la Tabla 3.1.

Dispositivo	**Ancho (px)**	**Alto (px)**
iPhone 6/7/8	375	667
iPhone 6/7/8 Plus	414	736
iPhone X	375	812
iPhone 11/XR	414	896
iPad	768	1024
iPad Pro	1024	1366
Galaxy S7	360	640
Galaxy S8/S9	360	740
Galaxy Tab	800	1280
Pixel 2	411	731
Pixel 2 XL	411	823
Nexus 5X/6P	1080	1920
Nexus 9	768	1024
G5	1440	2560

Tabla 3.1 Tamaños de pantalla según Vuforia Studio.

Una vez obtenidos los datos, se establecieron cuatro categorías de pantalla: estrechas, medianas, amplias y muy amplias, cubriendo todos los dispositivos disponibles en la `Vista previa`. La implementación de esta clasificación, junto con la orientación del dispositivo, se realizó mediante el anidamiento de los intervalos de tamaño de pantalla dentro de las condiciones de orientación. De este modo, se definió un fragmento para la orientación vertical que abarca todos los tamaños de pantalla y otro para la orientación horizontal que cumple la misma función, generando una estructura similar a la mostrada en el Código 3.4.

```
//-------------Orientacion horizontal----------------//
@media (orientation: landscape) {                          // Estilos aplicados en la orientacion horizontal
                                                           // evalua en que intervalo cabe el display actual
  @media(max-width: 580px){ /*parametro: valor1*/ }        // Pantallas Estrechas (Menos de 580 px de ancho)
  @media(min-width: 581px) and (max-width: 736px){/*parametro: valor2*/}          // Pantallas Medianas
  @media(min-width: 737px) and (max-width: 1024px){/*parametro: valor3*/}         // Pantallas Amplias
  @media(min-width: 1025px) {/*parametro: valor4*/}        // Pantallas Muy Amplias (Mas de 1024 px de ancho)
}
```

Código 3.4 Estructura del diseño adaptativo implementado con *media queries*.

Cabe aclarar que el parámetro modificado es el mismo en todos los intervalos; lo único que varía es el valor asignado. Además, los parámetros modificados dentro de los *media queries* deben estar dentro de una clase de CSS, es decir, la estructura presentada debe contener clases, con parámetros y valores, correctamente codificados, tal como se encuentra en el Código 3.3.

A continuación, se presenta un ejemplo del uso de este enfoque durante el diseño de la aplicación: partiendo de la portada parcialmente terminada, cuya estructura se puede ver en la Figura 3.4, se agregó una imagen y un texto sin asignarles ninguna clase ni modificar sus propiedades, dando como resultado la Figura 3.7a.

Para ajustar la proporción de la `vista`, es posible modificar los valores de las propiedades de la imagen en el panel de `propiedades`; sin embargo, el tamaño de la letra en las etiquetas no se puede ajustar de la misma manera. Por esta razón, se optó por estandarizar las modificaciones de tamaño usando CSS para ambos elementos, creando la clase mostrada en el Código 3.3 para la etiqueta y la clase `.P_plantaH` (mostrada en el Código 3.5) para la imagen.

```
.P_plantaH{                      // Clase que sera asignada al renderizado de la planta
   width: 75vw;                  // Ancho de la imagen, establecido en unidades de ancho respecto a la pantalla
   margin-top:5vh;               // Margen superior, agregado para separar el titulo de la imagen
   height: auto !important;      // Sentencia para mantener la proporcion de la imagen si cambia el ancho
}
```

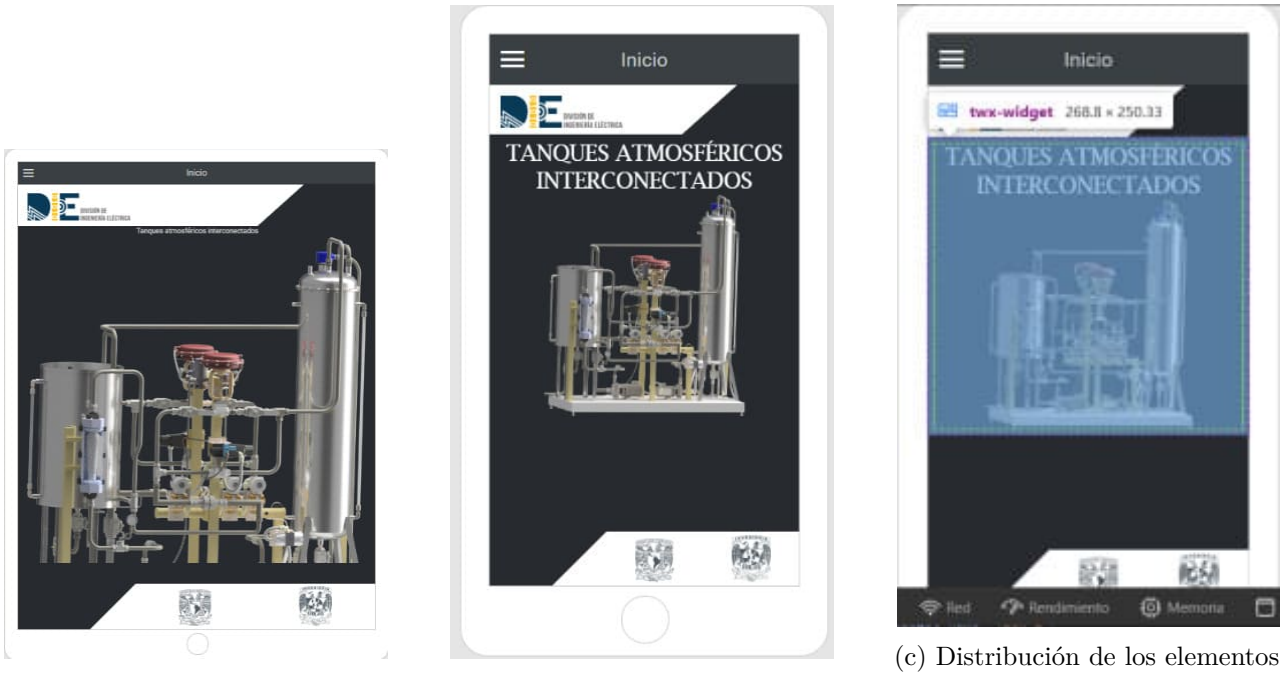
Código 3.5 Clase en CSS para la imagen de la portada.

Usando la consola de desarrollo, tal como se ha descrito anteriormente, se modificó el valor del ancho de la imagen a un tamaño adecuado; en este caso, un valor de `65vw` aprovecha bien el espacio en la mayoría de los dispositivos. El mismo procedimiento se aplicó al texto, determinando que un valor de `7vw` era el más apropiado; tras aplicar los cambios, el resultado fue similar a la Figura 3.7a.

Aunque el tamaño es adecuado, hay dos problemas: la imagen de los tanques está muy cerca del título y los objetos no están centrados verticalmente. Para corregir lo primero, se añadió un margen superior de `5vh` en la clase asignada a la imagen; las unidades `vh` son ideales para este ajuste, ya que garantizan que la separación vertical entre el título y la imagen sea siempre del 5 % de la altura de la pantalla, independientemente del dispositivo.

Para el segundo problema, a pesar de que las propiedades del contenedor central (`PanelCentral`) y del diseño de cuadrícula indican que el contenido debería estar centrado, esto no ocurre. El motivo es que la altura del `PanelCentral` está ajustada al tamaño de su contenido; es decir, el contenedor central tiene la altura correspondiente a la imagen y al título, y no la altura del espacio disponible entre los trapezoides blancos (Figura 3.7c). Por lo tanto, el contenido se encuentra centrado respecto al tamaño de su contenedor, pero no con respecto al espacio total disponible en la pantalla.

Analizando la estructura del diseño (Figura 3.4), se observa que el contenedor central debe tener una altura equivalente a la distancia entre los dos bordes blancos; si esto es así y si las propiedades de alineación están configuradas correctamente, entonces el contenido siempre estará centrado respecto a la pantalla. Con la consola de desarrollo, se observó que para la mayoría de los dispositivos, este



(a) Portada sin escalas adecuadas. (b) Portada con clases asignadas. (c) Distribución de los elementos de la portada.

Figura 3.7 Corrección de problemas de posicionamiento mediante *media queries*.

espacio mide `72vh`, por lo cual la altura podría fijarse en este valor desde el panel de propiedades del `PanelCentral`.

Sin embargo, en dispositivos de pantalla grande, la altura del panel central debería ser mayor; esto se debe a que, aunque los trapezoides superior e inferior mantienen valores constantes (`9vh` cada uno), el menú de navegación ocupa un espacio variable y se considera dentro de la altura de la `vista`. En la mayoría de los dispositivos, el espacio ocupado por este menú es de `10vh`, pero en dispositivos grandes es de unos `4.6vh`.

En este punto interviene el diseño adaptativo mediante *media queries*, ya que es necesario ajustar el valor de una propiedad en función de la resolución de la pantalla. Para abordar el problema descrito, se definió el valor adecuado de la altura del contenedor central según la resolución correspondiente, dentro de los *media queries* respectivos. Para ello, se creó una clase denominada `P_centralPane`, la cual fue asignada al *widget* `PanelCentral`. Al definir los valores apropiados en cada caso, se obtuvo el resultado mostrado en el Código 3.6.

```
// ====================== ORIENTACION VERTICAL ========================== //
@media (orientation: portrait) {            // NOTA: los estilos definidos fuera de los intervalos se aplican
                                            // en todos los intervalos de orientacion vertical a menos que
                                            // haya un valor especificado dentro de un media querie
  .P_plantaH {                              // Estilo para la imagen en orientacion vertical
    width: 65vw;                            // Ancho de la imagen en viewport width (vw)
    height: auto !important;                // Altura automatica para mantener la proporcion
    margin-top: 5vh;                        // Margen superior de 5vh para separar la imagen del texto
  }
  .P_title-label{                           // Estilo para el texto en orientacion vertical
    font-size: 7vw;                         // Tamano de fuente en vw para adaptarse a diferentes resoluciones
  }
  .P_centralPane{                           // Estilo para el panel central en orientacion vertical
    height: 72vh;                           // Altura del panel central, adaptada para la mayoria de dispositivos
    padding: 0;                             // Sin padding para asegurar que el contenido ocupe todo el espacio
  }
  @media (min-width: 509px){                // Solo dentro de este intervalo se cambia el valor de la altura
    .P_centralPane{height: 76.4vh;}         // La altura se calculo con la operacion: 100-18-(4.6)
  }
```

```
20 }
```

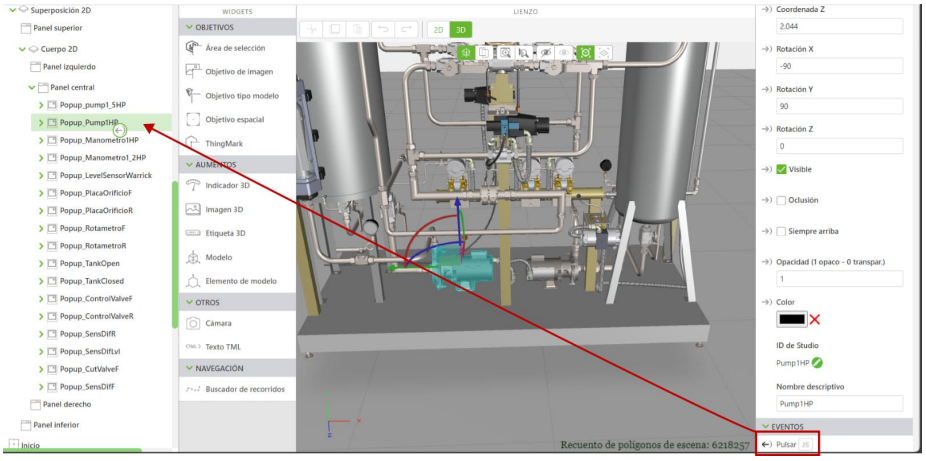
Código 3.6 Implementación de diseño adapatativo con *media queries*.

En la experiencia desarrollada, este principio también se aplicó para definir las escalas de menús emergentes y botones. Para hacerlo, se crearon los menús o botones ajustando sus proporciones usando unidades fijas (ver Tabla 2.9) en un dispositivo grande como el iPad; este fue el menú a escala 1. Posteriormente, se definió el origen de la transformación de escala, que dependió de la ubicación deseada del botón o menú, y finalmente, se agregaron las transformaciones de escala en los *media queries* correspondientes al tamaño de la pantalla. Los valores adecuados de escala para cada dispositivo fueron conseguidos con el método descrito en la subsección 3.2.1.

3.3. Uso de eventos en Vuforia Studio

Durante el desarrollo de la experiencia de AR, se utilizaron algunos `eventos` de los *widgets* para enlazar acciones de los usuarios con comportamientos dentro de la aplicación. En el entorno de Vuforia Studio, estos comportamientos se denominan `servicios`; entre los más usados se encuentran: `visibilidad de objetos`, saltar entre `vistas`, `mostrar u ocultar` ***`pop-ups`***, entre otros.

Las relaciones que existen entre `eventos`, `parámetros` de la aplicación y `servicios` o `propiedades` se denominan `enlaces`. Gracias al modelo *drag-and-drop* es posible crear `enlaces` arrastrando el ratón de la computadora personal (PC, por sus siglas en inglés) con el *clic* izquierdo sostenido, desde el símbolo de enlace (ubicado del lado izquierdo del evento deseado) hacia el árbol del proyecto, y soltando el botón sobre el *widget* que se desea modificar, tal como en la Figura 3.8a. Al hacerlo, se despliega un menú llamado `Seleccionar destino de enlace`, en el cual se muestran los servicios que es posible enlazar; un ejemplo se muestra en la Figura 3.8b. Para consultar los eventos y servicios disponibles de un *widget* en particular, basta con desplazarse a la parte inferior del panel de propiedades del mismo.



(a) Enlace de eventos en la UI.

(b) Menú de eventos disponibles para un *widget*.

Figura 3.8 Enlace de eventos con servicios *widgets* en Vuforia Studio. [7]

De la misma forma, este proceso se puede realizar con `parámetros de la aplicación`; estas son variables que el desarrollador puede definir desde el panel de `PARÁMETROS DE APLICACIÓN`, que

[7] Figura de autoría propia.

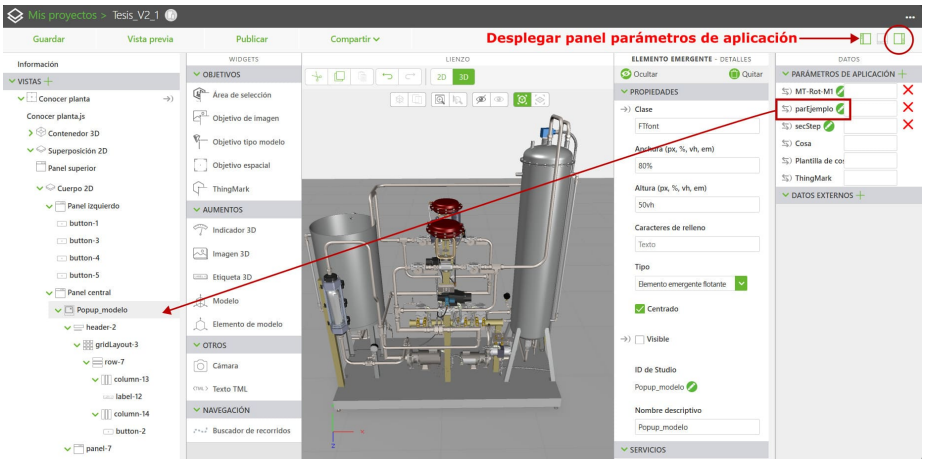
se despliega con el botón correspondiente del lado derecho (Figura 3.9a). Al crear y nombrar un parámetro, se puede realizar el mismo proceso de enlace descrito anteriormente, con la diferencia de que el menú desplegado permite enlazar a `propiedades` de *widgets*, no a `servicios` (ver Figura 3.9b).

Además, los cambios en las propiedades de los *widgets* también pueden realizarse mediante código en JavaScript, accediendo a ellas a través del `ID de Studio` utilizando notación de corchetes. Por ejemplo, en la función presentada en el Código 3.7 se emplea un parámetro de aplicación denominado `ToggleOper`, el cual adopta el valor de un botón de alternar con `ID de Studio ToggleOpen`. Posteriormente, `ToggleOper` modifica la visibilidad de un *pop-up* con `ID de Studio Oper_M2`. Como resultado, en la `vista` se obtuvo un botón de alternar que permite mostrar u ocultar un menú según su estado.

```
$scope.alternarMenu = function(){          // Funcion para laternar la visibilidad de un pop-up llamado Oper_M2
   $scope.app.params.ToggleOper=$scope.view.wdg['ToggleOpen']['value'];  // El parametro toma el valor del boton
   $scope.view.wdg['Oper_M2']['visible']=$scope.app.params.ToggleOper;   // El parametro controla la visibilidad
}
```

Código 3.7 Uso de parámetros de aplicación en JavaScript.

La función debe ser llamada a través de un evento de un *widget*. Esto se realizó escribiendo el nombre de la función en el apartado correspondiente, ubicado en el lado opuesto a la flecha utilizada para enlazar eventos, como se muestra en la Figura 3.8a, dentro de un recuadro gris identificado con las letras `JS`. De esta forma, cada vez que ocurrió el evento seleccionado, la función se ejecutó.



(a) Enlace de propiedades en la UI.

(b) Menú de propiedades disponibles para un *widget*.

Figura 3.9 Enlace de parámetros de aplicación con propiedades de *widgets*. [8]

El uso de enlaces permitió que la experiencia desarrollada tenga un alto nivel de interactividad, ya que la mayoría de los elementos visuales responden a la interacción del usuario, siendo posible desplazarse a través de *digital overlays* y `vistas` dentro de la aplicación, así como desplegar y ocultar menús emergentes que incluyen botones para controlar las animaciones y proporcionar información que facilita la comprensión del sistema de tanques. Asimismo, el uso de `parámetros de aplicación` enlazados a propiedades de *widgets* y ejecutados mediante funciones constituyó una de las herramientas principales para la creación de animaciones en modelos CAD. En las siguientes subsecciones se describe la implementación de estas funcionalidades.

[8] Figura de autoría propia.

3.3.1. Botones de navegación

La experiencia de AR desarrollada para el sistema de tanques atmosféricos interconectados incluye siete `vistas`, entre las que se encuentran una portada y diversas pantallas con modelos CAD del sistema completo y de algunos equipos individualmente. Sobre estos modelos se superponen animaciones de funcionamiento, tarjetas con datos relevantes, así como despieces y secuencias operativas.

Para garantizar una experiencia de uso intuitiva y sencilla, se decidió integrar todos los *digital overlays* correspondientes a un mismo modelo 3D dentro de una sola `vista`. Específicamente, en la pantalla que muestra el modelo tridimensional del sistema de tanques, se incorporaron el *digital overlay* denominado `Operación`, que muestra los circuitos de transferencia y recirculación, y el *digital overlay* denominado `Capacitación`, que detalla la instrumentación del sistema.

El primer reto para implementar esta filosofía de navegación consistió en sincronizar la visibilidad de cada *digital overlay* dentro de una sola `vista`, de modo que, al seleccionar la opción de `Capacitación`, no se visualizaran simultáneamente las animaciones de `Operación`. Para resolver este problema, se implementaron los conceptos de enlaces en las tres formas descritas anteriormente.

La `vista "Conocer planta"` despliega un menú en el lado izquierdo de la pantalla que incluye botones para seleccionar el *digital overlay* de `Operación` o el de `Capacitación` (Figura 3.10a). Al seleccionar `Operación`, se desencadenan dos procesos: en primer lugar, se despliegan los botones necesarios para controlar las animaciones correspondientes; en segundo lugar, durante la ejecución del código, se activan las variables que permiten habilitar únicamente las funcionalidades requeridas para esta opción, evitando interferencias con el *overlay* de `Capacitación`. Este segundo proceso se describe en detalle en la sección 3.3.3.

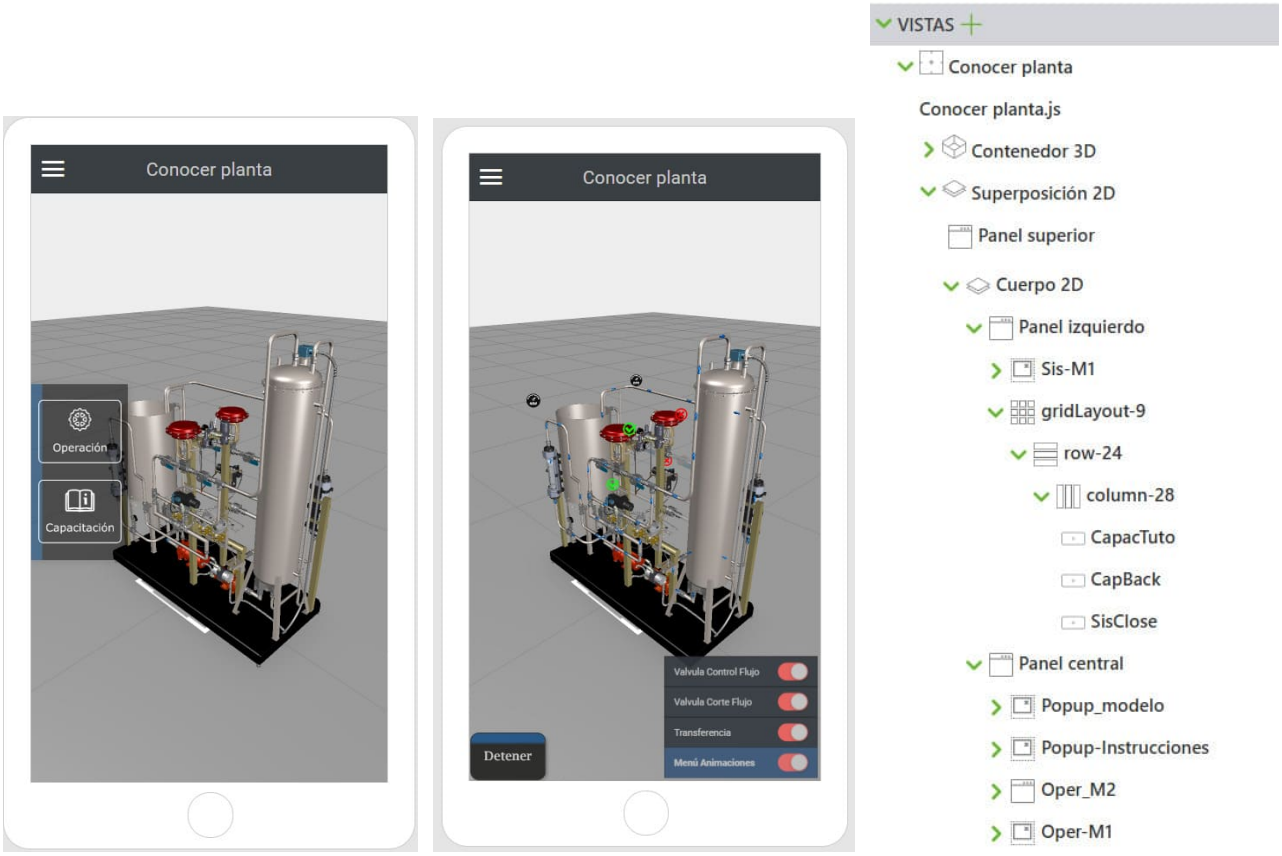
Para el primer proceso, se ejecutan las líneas de código contenidas en la función `ControlesOperacion()` (ver Código 3.8). Esta función enlaza las propiedades de visibilidad del panel `Oper_M2`, ubicado en la esquina inferior derecha de la Figura 3.10b, así como del botón `SisClose` y del *pop-up* `Oper-M1`, con el evento "Pulsar" del botón de `Operación`. Es importante destacar que cualquier cambio en la visibilidad de un contenedor afecta también a todos los elementos contenidos en su interior.

Asimismo, dentro de `Oper_M2` se encuentra un *widget* de alternar etiquetado como "Menú animaciones". Este *widget* permite mostrar u ocultar los botones de operación de las válvulas del sistema según la elección del usuario, mejorando la visibilidad del modelo cuando es necesario. Para implementar este comportamiento, se ejecuta la función `VisOperM1()`, la cual sigue una lógica similar a la presentada en el Código 3.8.

```
/*Funcion para habilitar los controles y animaciones del overlay de operacion*/
$scope.ControlesOperacion = function() {
  $scope.view.wdg['Sis-M1']['visible'] = false;                //Oculta el panel izquierdo (botones oper y mant)
  $scope.view.wdg['SisClose']['visible'] = true;               //Habilita el boton de cerrar menu de operacion
  $scope.view.wdg['Oper_M2']['visible'] = true;
  $scope.view.wdg['ToggleOpen']['visible'] = true;             //Hace visible el boton toggle del panel Oper-M1
  $scope.view.wdg['ToggleOpen']['value'] = true;               //El valor de inicio de ToggleOpen es true
};
$scope.VisOperM1 = function() {     // Funcion para ativar/ocultar los controles de valvulas con un boton toggle
  var visibility = $scope.view.wdg['ToggleOpen']['value'];     // La variable adopta el valor del boton
  $scope.view.wdg['CutVlvFlujo']['visible'] = visibility;      // La visibilidad alterna entre true y false, en
  $scope.view.wdg['CtrlVlvFlujo']['visible'] = visibility;     // funcion del valor del boton de alternar
  $scope.view.wdg['CutVlvRetro']['visible'] = visibility;      // esto sucede en todos los botones de valvulas
  $scope.view.wdg['CtrlVlvRetro']['visible'] = visibility;
  $scope.view.wdg['Switch-Circuits']['visible'] = visibility;
};
```

Código 3.8 Control de visibilidad de botones con JavaScript.

[9] Figura de autoría propia.



(a) Estado inicial de la vista. (b) *Digital overlay* de Operación. (c) Árbol de *widgets* 2D.

Figura 3.10 Funcionamiento y descripción de la vista `Conocer planta`. [9]

Finalmente, en el panel del árbol del proyecto mostrado en la Figura 3.10c, se puede ver un *widget* de `diseño de cuadrícula` que contiene tres botones. Los botones `SisClose` y `CapBack` se utilizan para regresar al estado inicial de la `vista` mostrado en la Figura 3.10a; esto se realiza mediante enlaces de tipo *drag-and-drop*, que conectan el evento `Pulsar` de ambos botones con el servicio `show pop-up` de `Sis_M1`. De manera similar, el botón `CapacTuto` está enlazado al servicio `show pop-up` de `Popup-Instrucciones`.

3.3.2. *Pop-ups*

Los *pop-ups* o elementos emergentes son *widgets* que funcionan como ventanas emergentes en las `vistas`; se diferencian de otros contenedores por los servicios con los que cuenta: mostrar y ocultar el elemento emergente. Esta flexibilidad permite manipular directamente la visibilidad de la ventana y los elementos contenidos sin necesidad de crear parámetros de aplicación.

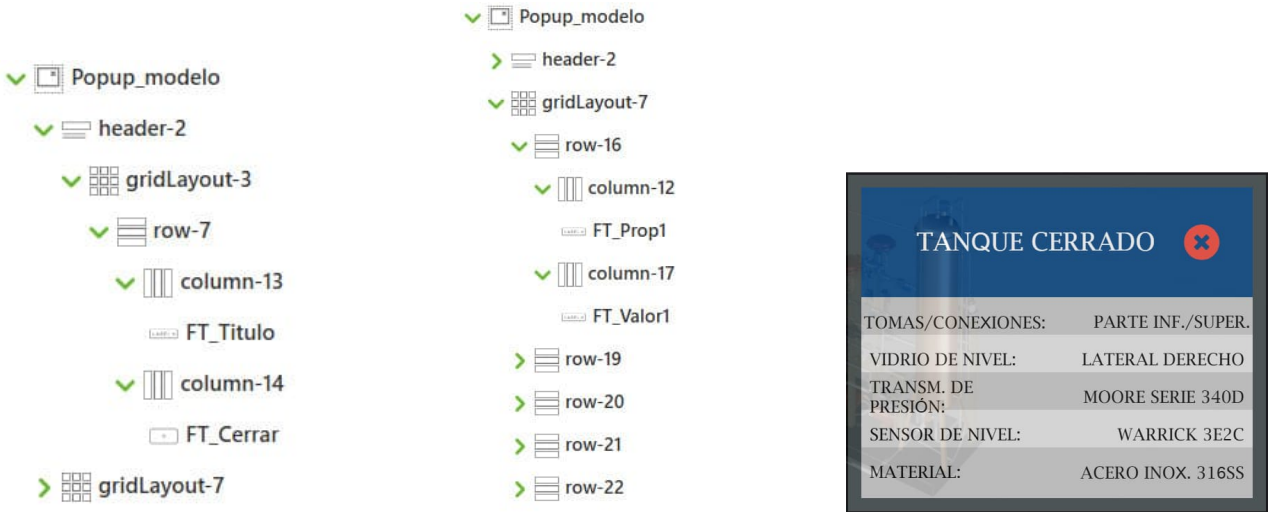
En la experiencia desarrollada, estos *widgets* se utilizan para crear los menús de operación, como los descritos en la sección anterior, así como las ventanas emergentes de instrucciones y las tarjetas informativas que describen la instrumentación del sistema. Para realizarlo, se usan los *pop-ups* como contenedor de los elementos que organizan la información.

Las tarjetas informativas constituyen el ejemplo más completo para ilustrar este concepto, ya que integran todos los principios usados para crear el resto de ventanas emergentes de la aplicación. Estas

tarjetas se encuentran en la opción `Capacitación` de la `vista Conocer planta` y tienen el objetivo de que el usuario visualice una ficha técnica de un instrumento en particular al tocarlo con el dedo.

El proceso de creación consistió en arrastrar un *pop-up* al lienzo 2D del proyecto; este elemento funge como el contenedor principal de toda la información y los *widgets* necesarios para organizarla. A continuación, se agregó un `panel` para el encabezado, con `ID header-2`, y una cuadrícula con `ID gridLayout7` para el contenido de la tarjeta, esto con el objetivo de mostrar la información de forma organizada y modificarla fácilmente en tiempo de ejecución.

La estructura consiste en un encabezado, construido por un diseño de cuadrícula con dos columnas: una para el nombre del instrumento y otra para el botón de cerrar (Figura 3.11a). Para el cuerpo de la tarjeta se utilizó otra cuadrícula, con cinco filas de dos columnas cada una; en la columna izquierda se colocaron las etiquetas para las propiedades del instrumento (`FT_Prop`) y en la derecha, las etiquetas para los valores de cada propiedad (`FT_Valor`). Esto se muestra en la Figura 3.11b. Los nombres de las etiquetas sirven para identificar los renglones de la tarjeta y modificar el texto de cada uno desde código JavaScript de forma sencilla y organizada.



(a) Estructura del encabezado. (b) Estructura del cuerpo. (c) Vista durante la ejecución.

Figura 3.11 Organización y funcionamiento del *pop-up* de ficha técnica. [10]

La visibilidad y la información del *pop-up* de la tarjeta informativa se modifican por código `JS` en el evento `Pulsar` de cada *widget* `elemento de modelo` que contiene al instrumento cuya información se desea mostrar. Con ese evento se llama a la función `mostrarPopup('nombreModelo')`, que modifica la propiedad de visibilidad del *pop-up*, así como el texto de las etiquetas, siguiendo la lógica mostrada en el Código 3.9. Conviene señalar que el fragmento de código modifica una pareja de propiedades y valores, de cinco que se encuentran en la tarjeta (Figura 3.11c).

```
var informacionModelo = {          //Objeto de objetos, contiene la info de toda la instrumentacion de la planta
  'BOMBA 1.5 HP' : {                                    // Sub-objeto de una bomba, conformado por 5 arreglos
    'renglon1' : ['Modelo:',              '4IME0150A'],     // Cada arreglo corresponde a una fila de la tarjeta
    'renglon2' : ['Potencia del motor:',  '1.5Hp'],         // La posicion 0 corresponde a una propiedad
    'renglon3' : ['RPM del motor:',       '3454 RPM'],      // La posicion 1 corresponde a una propiedad
    'renglon4' : ['Voltaje:',             '220/440 V'],     // Por lo tanto hay 5 parejas de propiedad-valor
    'renglon5' : ['Fases del motor:',     'Trifasico'],     // Solo se muestra 1 sub-objeto de 17 existentes
  },
};

```

[10] Figura de autoría propia.

```
$scope.mostrarPopup = function(nombreModelo){
  if(mostrarPopup){                                    // Evalua si el bit mostrarPopup esta activo
    $scope.view.wdg['FT_Titulo']['text'] = nombreModelo;    // El titulo es el ID del instrumento tocado
    $scope.view.wdg['FT_Prop1']['text'] = informacionModelo[nombreModelo]['renglon1'][0];  // Propiedad 1
    $scope.view.wdg['FT_Valor1']['text'] = informacionModelo[nombreModelo]['renglon1'][1]; // Valor 1
    $scope.view.wdg['FT_Prop2']['text'] = informacionModelo[nombreModelo]['renglon2'][0];  // Propiedad 2
    $scope.view.wdg['FT_Valor2']['text'] = informacionModelo[nombreModelo]['renglon2'][1]; // Valor 2
    $scope.view.wdg['Popup_modelo']['visible']=true;        // Tarjeta informativa visible
  }
};
```

Código 3.9 Código para controlar la información y visibilidad de las tarjetas informativas.

Para contener y mostrar toda la información de los instrumentos que conforman al sistema, se creó el objeto `informacionModelo`, conformado por diecisiete subobjetos que contienen la información de cada instrumento en cinco arreglos, correspondientes a cada renglón de la tarjeta informativa. Dichos arreglos constan de dos posiciones, en las cuales se encuentran las propiedades y sus respectivos valores. Después, se definió la función `mostrarPopup`, cuyo parámetro corresponde al `ID de Studio` del instrumento que se desea destacar. Este parámetro tiene dos usos: colocarse como título de la tarjeta e identificar qué subobjeto de `informacionModelo` debe ser usado para obtener los textos de la tarjeta informativa. Además, la función usa las posiciones de los arreglos para mostrar cada propiedad en la columna izquierda y sus valores en la columna derecha.

Esta lógica permite usar una sola función y un solo *pop-up* para mostrar la información de cualquier instrumento, reduciendo los elementos en el lienzo 2D y permitiendo agregar o modificar información del sistema de forma sencilla.

3.3.3. Funciones

Como se ha visto a lo largo del capítulo, la mayoría de funciones son llamadas a través de `eventos` de *widgets*; sin embargo, también es posible ejecutarlas desde otras funciones en JavaScript. Esta capacidad se usó para ejecutar muchas funciones en un solo fragmento de código, optimizar tareas repetitivas, organizar las acciones realizadas y programar animaciones.

En la presente sección se pretenden explicar algunos de los usos extendidos de las funciones, así como entrar más en detalle sobre su sintaxis y algunos otros aspectos de las mismas, lo cual servirá como preámbulo para la siguiente sección del desarrollo.

Actualizaciones en la vista

La función `verificarEstados` se diseñó para actualizar las animaciones según la interacción del usuario, gestionando la visibilidad de diversos objetos en la `vista Conocer Planta` bajo el modo `Operación`. Este *overlay*, ilustrado en la Figura 3.10b, integra *widgets* `3DGauge` con indicadores rojos y verdes que señalan el estado de las válvulas en los circuitos de transferencia y realimentación. Asimismo, los flujos de agua se representan mediante flechas azules definidas como `elementos de modelo`; su visibilidad se sincroniza dinámicamente para mostrar únicamente aquellas que corresponden al circuito seleccionado.

Cuando el usuario elige un circuito, el sistema aplica un efecto de transparencia (0.5) a las tuberías del circuito contrario para darle prioridad visual al seleccionado. Los controles de las válvulas se alternan según esta elección; así, el usuario solo interactúa con los mandos del circuito visible. Este diseño responde a la necesidad de que, al desactivar un flujo, las válvulas del otro circuito se abran para evitar obstrucciones.

La sincronización se gestiona a través de la función `verifcarEstados`, la cual toma en cuenta los estados de todas las válvulas del sistema, asignándoles una variable booleana enumerada de la letra A a la E. Además, la visibilidad de todos los indicadores también depende del *bit* `detenerAnimacionAgua`, el cual garantiza que los elementos del *overlay* de `Operación` no sean visibles si el usuario selecciona la opción de `Capacitación`.

La función completa realiza los procesos de la lista presentada a continuación. Al acabar esas tareas, actualiza los cambios en la `vista` usando el método `$scope.$apply()`.

1. Declara las variables necesarias para controlar la visibilidad de las flechas, los indicadores de las válvulas, la opacidad de los circuitos y los botones para cambiar el estado de las mismas.

2. Determina qué flechas serán visibles, dependiendo del circuito seleccionado por el usuario: las flechas 1 a 11 siempre son visibles porque, para ambos casos, el flujo de agua siempre empieza en el primer tanque y pasa por la bomba de 1.5 HP. Después de la primera bifurcación, las flechas que corresponden al circuito de transferencia son de la 12 a la 41, y el resto pertenecen al circuito de recirculación.

3. Determina la visibilidad de los controles para apagar y encender válvulas, tomando en cuenta la lógica descrita anteriormente y el valor del botón para ocultar los controles de las válvulas (var F).

4. Controla la visibilidad de los indicadores de estado de las válvulas, basándose en sus estados.

El Código 3.10 muestra una versión acotada de esta función, para demostrar la lógica de funcionamiento.

```
var verificarEstados = function(bool){
  var A = detenerAnimacionAgua;
  var B = estadoCutValveF;
  var C = estadoControlValveF;
  var D = estadoCutValveR;
  var E = estadoControlValveR;
  var F = TO;
  //Operaciones booleanas para determinar la visualizacion de los circuitos
  var flechaAguaN;
  for(var j = 1; j <= 55; j++){  // La variable "flechaAgua-" adopta valores de "flechaAgua-1" a "flechaAgua-55"
    flechaAguaN = 'flechaAgua-' + j.toString();
    /* (Flechas que bajan del tanque uno y llegan hasta la primer bifurcacion*/
    if(j >= 1 && j <= 11){$scope.view.wdg[flechaAguaN]['visible'] = !A;}
    /* Son las flechas que componen el CIRCUITO DE TRANSFERENCIA despues de la bifurcacion
       cuando las valvulas de control y corte del flujo circuito estan abiertas se muestra Trans*/
    else if(j >= 12 && j<= 41){$scope.view.wdg[flechaAguaN]['visible'] = !A && !(B && C);}
    /* Son las flechas que componen el CIRCUITO DE RETROALIMENTACION despues de la bifurcacion
       cuando las valvulas de recirculacion estan abiertas, se muestra realimentacion*/
    else{$scope.view.wdg[flechaAguaN]['visible'] = !A && !(D && E);}
  }
  //Operaciones aritmeticas para determinar la opacidad de los circuitos
  // Mientras Recirculacion este activado, la opacidad de Recir baja a 0.3
  $scope.view.wdg['circuitoTubTran']['opacity'] = 0.3 * !(D && E) + 1 * (!(B && C) || A);
  // Mientras Transferencia este activado, la opacidad de Recir baja a 0.3
  $scope.view.wdg['circuitoTubRetro']['opacity'] = 0.3 * !(B && C) + 1 * (!(D && E) || A);
  //Operaciones aritmeticas para determinar si estan deshabilitados los botones
  // Si el circuito de recirculacion esta activo, no se activan los controles de flujo
  $scope.view.wdg['CutVlvFlujo']['visible'] = F && !(!A && !(D && E));
  $scope.view.wdg['CtrlVlvFlujo']['visible'] = F && !(!A && !(D && E));
  // Si el circuito de flujo esta activo, no se activan los controles de retroalimentacion
  $scope.view.wdg['CutVlvRetro']['visible'] = F && !(!A && !(B && C));
  $scope.view.wdg['CtrlVlvRetro']['visible'] = F && !(!A && !(B && C));
  //Operaciones booleanas para determinar la visualizacion de los indicadores de las valvulas de control F
  $scope.view.wdg['CrossCtrlValveF']['visible'] = !A && C;     // Indicador de ON para la Ctrl Valve Flujo
  $scope.view.wdg['CheckCtrlValveF']['visible'] = !A && !C;    // Indicador de OFF para la Ctrl Valve Flujo
  //Operaciones booleanas para determinar la visualizacion de los indicadores de las valvulas de control R
  $scope.view.wdg['CheckCtrlValveR']['visible'] = !A && !E;    // Indicador de ON para la Ctrl Valve Recirc
```

```
38    $scope.view.wdg['CrossCtrlValveR']['visible'] = !A && E;      // Indicador de OFF para la Ctrl Valve Recirc
39    $scope.$apply();                                              // Actualiza los cambios en la vista
40  };
```

Código 3.10 Función para actualizar los cambios en la opción `Operación` de la Vista `Conocer Planta`

Las expresiones lógicas se derivaron mediante tablas de verdad y mapas de Karnaugh. Bajo la condición de animaciones habilitadas ($A = 0$), la opacidad del circuito disminuye ($f = 0$) únicamente cuando ambas válvulas están cerradas ($B = 1$ y $C = 1$). Este comportamiento se observa en la Tabla 3.2, donde las variables representan el *bit* de visibilidad y los estados de las válvulas de corte (`CutValveF`) y control (`ControlValveF`). De acuerdo con esta lógica, el circuito de transferencia se vuelve transparente solo si el usuario desactiva ambos componentes; en cualquier otro escenario (una o ninguna válvula presionada), el circuito permanece visible ($f = 1$). Este mismo principio rige la Tabla 3.3.

A	B	C	f
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	1

Tabla 3.2 Tabla de verdad para circuito de flujo.

A	D	E	f
1	1	1	0
1	1	0	0
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	1

Tabla 3.3 Tabla de verdad para circuito de recirculación.

Después de que se tomaron esos datos, se ordenaron en los mapas de Karnaugh 3.2 y 3.3 y con ellos se obtuvieron las expresiones booleanas que representan las condiciones para la visibilidad de los circuitos de transferencia (ecuación 3.1) y recirculación (ecuación 3.2).

$$f = !A\&\&!B||!A\&\&!C = !A\&\&(!B||!C) = !A\&\&!(B\&\&C) \tag{3.1}$$

$$f = !A\&\&!D||!A\&\&!E = !A\&\&(!D||!E) = !A\&\&!(D\&\&E) \tag{3.2}$$

A \ BC	00	01	11	10
0	1	1	0	1
1	0	0	0	0

Tabla 3.4 Grupos para visibilidad del circuito de flujo.

A \ DE	00	01	11	10
0	1	1	0	1
1	0	0	0	0

Tabla 3.5 Grupos para visibilidad del circuito de recirculación.

Secuencias

La creación de secuencias constituye un pilar fundamental de la experiencia desarrollada, ya que permite organizar cronológicamente las animaciones de piezas y maquinaria 3D, siguiendo una línea de tiempo específica con estados o pasos claramente diferenciados. De este modo, es posible representar fielmente comportamientos, procedimientos operativos o flujos de trabajo. Estas transformaciones abarcan desde movimientos de traslación, rotación y escalamiento hasta modificaciones en la apariencia visual, como el color y opacidad de los modelos CAD.

Además, las secuencias responden a las interacciones del usuario; en el caso de la experiencia desarrollada, se incluyen botones para pasar de un estado al siguiente, retroceder y reiniciar la secuencia completa. Esto permite que el usuario tenga control total sobre la sucesión de pasos, lo que agrega interactividad y facilidad para que las explicaciones se ajusten a la velocidad de entendimiento de cada usuario.

La `vista` de una secuencia típica luce como la Figura 3.12, incluye tres botones de control en la parte inferior, un panel para indicar el número de estado o paso en el que se encuentra el usuario y en la parte central el modelo tridimensional animado según la secuencia definida. Los botones y el indicador tienen clases CSS asignadas para mantener el mismo estilo visual que todas las otras `vistas`.

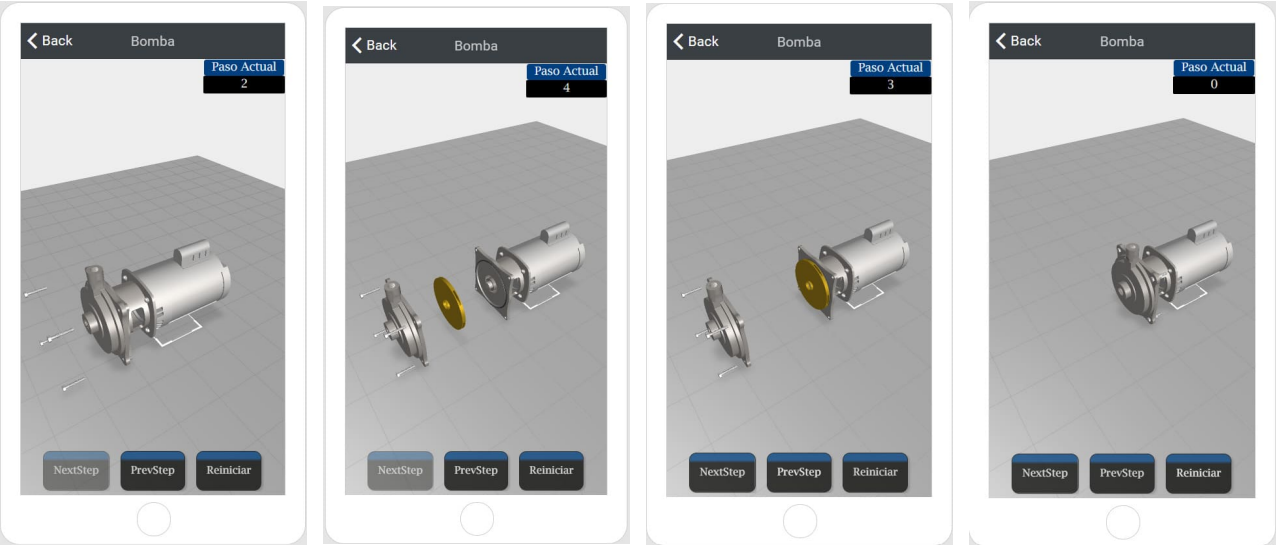
El evento `pulsar` de los botones de control ejecuta una función llamada `secuencia(step)`, como la mostrada en el Código 3.11, cuyo parámetro debe ser +1 o -1, el cual indica si el usuario desea que la secuencia avance al siguiente paso o repita el paso anterior. Los pasos o estados se definen dentro de la función mencionada, en una estructura *switch-and-case* que recibe como entrada el valor de un contador llamado `count`, inicializado en cero. Dentro de cada `caso` se ejecutan las animaciones en los modelos correspondientes a la secuencia deseada.

Si el usuario presiona el botón `NextStp`, se ejecuta la función `secuencia(+1)`. Dentro del código JavaScript, se suma el parámetro recibido a `count` para registrar el paso actual de la secuencia. Además, un condicional `if` detecta el signo del parámetro y ejecuta el *switch-and-case* correspondiente al orden ascendente de los pasos de la secuencia. Al hacer eso, se desactiva el botón `NextStep` hasta que la promesa correspondiente al paso actual se cumpla. Al llegar el contador al último paso de la secuencia, el botón `NextStp` permanece desactivado, hasta que se presione `PrevStp` o se reinicie la secuencia.

Para el botón de retroceder (`PrevStp`), el parámetro `step = -1` disminuye el valor del contador `count` y ejecuta el *switch-and-case* para el orden descendente de la secuencia. En estos `casos`, se llama a la función `Reset('nombreModelo')` para todos los modelos que se animaron en el paso actual. Es decir, después de ejecutarse un paso, el usuario tiene la opción de volver a observar las animaciones correspondientes a ese paso presionando el botón (`PrevStp`); esto devuelve todos los objetos a su estado anterior.

La función `Reset('nombreModelo')`, cuyo parámetro es el `ID de Studio` de un elemento 3D, devuelve el modelo a sus valores iniciales de color, posición, coordenadas y ángulo de rotación. Esto se realiza igualando los valores del objeto 3D a los valores almacenados en el objeto `parametrosModeloItem`. Este objeto contiene los valores iniciales de los parámetros de todos los modelos de la `vista` 3D, y se crea copiando el objeto `$scope.view.wdg` sin asignación de referencia, como se mostró en el Código 2.3.

Finalmente, el botón reiniciar llama a la función `reinicio()`, la cual ejecuta `Reset('nombreModelo')` para todas las piezas que conforman el modelo 3D animado. Además, establece el valor del contador `count` en cero, de forma que si se presiona el botón `NextStp`, la secuencia reproduce el paso 1.



(a) `NextStp` pulsado. (b) Último paso. (c) `PrevStp` pulsado. (d) `Reset` pulsado.

Figura 3.12 Ejecución de una secuencia completa. [11]

```
//----------------Secuencias-----------------------//
$scope.secuencia= function(step){
  count = count + step;                                          // Contador registra el paso actual
  if (step==+1){                                                 // Sentido ascendente de la secuencia
    $scope.view.wdg['NextStp']['disabled'] = true;               // Boton NextStp deshabilitado
    var promesas = [];                                           // Arreglo de promesas
    switch(count) {                                              // Count es evaluado para reproducir
      case 1:                                                    // las animaciones correctas. Paso 1
        promesas[0] = $scope.Traslacion('modeloTapa', 'y', 0.4, 60, '+'); // Promesa de la funcion traslacion
        promesas[1] = $scope.Parpadeo('modeloTapa', 70);         // Promesa de la funcion parpadeo
        break;                                                   // Termina el paso 1
      case 2:                                                    // Paso 2 con sus animacines definidas
        promesas[0] = $scope.Traslacion('Impulsor', 'y', 0.2, 60, '+');  // Promesa de la funcion traslacion
        promesas[1] = $scope.Rotacion('Impulsor', 'ry', 30, false);      // Promesa de la funcion rotacion
        break;                                                   // Termina el paso 2
    }
    Promise.all(promesas).then((resolves) => {                   // Promise.all espera que se cumplan todas promesas
      var activarBotonSecuencia = function(){                    // Se activa NextStep cuando se cumpla lo anterior
        if(count < 4){                                           // Siempre y cuando el contador sea menor a 4
          $scope.view.wdg['NextStp']['disabled'] = false;}       // Se habilita el boton NextStep
      };$timeout(activarBotonSecuencia, 10);                     // activarBotonsecuencia se ejecuta despues de 10ms
    }).catch((errores) => {console.log(errores);});              // catch() cuando al menos una promesa no se cumple
  }
  if (step==-1){                                                 // Sentido descediente de la secuencia
    $scope.view.wdg['NextStp']['disabled'] = false;              // Se habilita el boton NextStep
    switch(count) {                                              // Count es evaluado para resetear los modelos
      case 1:                                                    // Resets para volver al estado inicial del paso 1
        $scope.Reset('modeloTapa');                              // Devuelve modeloTapa a su coordenada Y original
        count=0;                                                 // Contador es el paso reseteado -1
        break;
      case 2:                                                    // Resets para volver al estado inicial del paso 2
        $scope.Reset('Impulsor');                                // Devuelve Impulsor al estado inical del paso 2
        count=1;                                                 // Contador es el paso reseteado -1
        break;
    }
  }
  if (count>4){count=4}                                          // El contador no puede tener un valor mayor a 4
  if (count<0){count=0}                                          // El contador no puede tener un valor menor a 0
```

[11] Figura de autoría propia.

```
39    $scope.app.params.secStep = count;                    // Se actualiza el numero en el panel de paso actual
40  }
```

Código 3.11 Función para definir y reproducir las animaciones de una secuencia.

3.4. Animaciones en JavaScript

Al observar experiencias de AR en diversas fuentes, incluidos los proyectos de muestra que proporciona Vuforia Studio, es evidente que la mayoría de las secuencias animadas relacionadas con montaje, mantenimiento, operación y reparación son combinaciones de animaciones básicas aplicadas a piezas específicas de las instalaciones industriales.

Esto significa que, para crear secuencias complejas de procesos operativos sin recurrir a *software* adicional, se necesitan tres elementos: piezas individuales que componen una máquina o instalación industrial, animaciones básicas aplicadas a dichas piezas y una forma de organizar la ejecución de las animaciones siguiendo una línea de tiempo definida.

En la sección anterior se explicó una forma para determinar cómo y cuándo ejecutar funciones, con lo que se cubre el tercer elemento.

Para el segundo elemento, Vuforia Studio ofrece un *widget* llamado `elemento de modelo`, que permite seleccionar una parte específica del modelo CAD cargado y aislarla para aplicar propiedades solo a esa pieza, sin afectar al resto del modelo [PTC, 2024a].

Por lo tanto, el desafío consistió en crear animaciones que movieran, rotaran, ocultaran o destacaran piezas individuales o conjuntos funcionales. Aprovechando las propiedades del *widget*, junto con servicios, eventos y propiedades comunes, se crearon funciones que modifican parámetros como coordenadas, ángulos de rotación, color, opacidad o visibilidad, ya sea de manera abrupta o continua. Al combinar y sincronizar varias de estas animaciones, se pueden obtener secuencias muy similares a las generadas por *software* especializado.

Una vez definidas las animaciones básicas para crear secuencias complejas, se buscó dotar a las funciones de la mayor flexibilidad posible, de modo que pudieran aplicarse en diversas circunstancias y cubrir todas las necesidades de animación. Para ello, se definió que las funciones requieran varios parámetros para funcionar, los cuales determinan características como la velocidad y tiempo de ejecución, la dirección, el sentido, entre otros.

En esta sección se presentan las soluciones implementadas para cada tipo de animación, describiendo cómo se utilizan y destacando los aspectos clave de su lógica de programación.

3.4.1. Traslación

La primera animación fundamental es la de **traslación**, la cual se ejecuta cuando se desea mover un objeto en un eje específico dentro del espacio 3D. La función fue diseñada para ser flexible, pues permite mover un `elemento de modelo` en cualquiera de los ejes principales (`X, Y` o `Z`), así como definir la velocidad del movimiento. Además, incluye verificaciones para evitar movimientos no deseados o infinitos cuando se vuelve a llamar la función durante el tiempo de ejecución.

La función, llamada `Traslacion(nombreModelo, eje, Avance, timingInterval, sentido)`, recibe como parámetros:

- `NombreModelo`: `ID de Studio` del elemento que se desea mover.

- `Eje:` El eje en el que se desea realizar la traslación (`X, Y` o `Z`).
- `Avance:` La distancia máxima que se permitirá mover el objeto desde su posición original.
- `TimingInterval:` Intervalo de tiempo en milisegundos que define la frecuencia con la que se actualizará la posición del objeto.
- `Sentido:` La dirección del movimiento, ya sea positivo o negativo respecto al eje seleccionado.

Cuando se ejecuta la función, esta inicia una promesa que controla la ejecución asíncrona de la animación. En primer lugar, la función obtiene la coordenada original del modelo en el eje especificado a través del objeto `parametrosModeloItem`; con esta información, evalúa si la coordenada actual del modelo no coincide con la original. Si detecta que las posiciones no coinciden, devuelve el objeto a su posición inicial, evitando que se genere un movimiento infinito cuando el botón se presiona repetidamente.

Luego, se inicializa una variable llamada `estado`, que se utiliza para monitorear el progreso de la animación. A continuación, se define un ciclo repetitivo con `$interval`, que actualiza la posición del modelo a intervalos de tiempo regulares. Dentro de este ciclo se evalúan tres escenarios:

1. Si el modelo está en su coordenada original y su nombre se encuentra en un arreglo llamado `listaObjetosTranslacion`, esto indica que el botón ha sido presionado nuevamente. En ese caso, se cancela la animación y la variable `estado` se establece nuevamente en verdadero.
2. Si `estado` es falso y el objeto no se encuentra en `listaObjetosTranslacion`, significa que la animación de movimiento no se está ejecutando en ese momento. Entonces, el objeto se agrega a la lista de objetos en translación y su posición se actualiza aplicando un pequeño avance.
3. Si las coordenadas del modelo exceden el avance definido por el parámetro `avance`, se cancela el intervalo, lo que detiene la animación. Además, se elimina el objeto de la lista de objetos en traslación.

Esta estructura asegura que la animación funcione de manera controlada y que no se superpongan movimientos indeseados al llamar muchas veces la función. Una versión explicativa de esta se encuentra en el Código 3.12.

```
$scope.Traslacion = function(nombreModelo, eje, Avance, timingInterval, sentido) {
  return new Promise((resolve, reject) => {
    var cordenadaOriginal = parametrosModeloItem[nombreModelo][eje];  // Guardar posicion original
    if ($scope.view.wdg[nombreModelo][eje] != cordenadaOriginal) {    // Si no esta en posicion original
      $scope.view.wdg[nombreModelo][eje] = cordenadaOriginal;         // Restablecer posicion original
    }
    var updateCordenada = $interval(function() {                      // Iniciar movimiento con intervalo
      if ($scope.view.wdg[nombreModelo][eje] == cordenadaOriginal) {  // Verificar si boton fue presionado
        $interval.cancel(updateCordenada);                            // Detener movimiento
        reject('Se presiono el boton de nuevo');                      // Rechazar promesa
      }
      $scope.view.wdg[nombreModelo][eje] += Number(sentido) * 0.006;  // Mover objeto
      if (Math.abs($scope.view.wdg[nombreModelo][eje] - cordenadaOriginal) >= Avance) {
        $interval.cancel(updateCordenada);                            // Detener al alcanzar limite
        resolve('Operacion de traslacion exitosa');                   // Resolver promesa
      }
    }, timingInterval);                                               // Intervalo de tiempo
  });
};
```

Código 3.12 Función simplificada para realizar animación de traslación.

3.4.2. Rotación

La función de rotación permite girar un objeto en torno a uno de sus ejes. Esta animación se logra mediante la actualización continua del ángulo de rotación del objeto, controlada por parámetros definidos como el número total de vueltas y la visibilidad del objeto una vez finalizado el movimiento. Esto último fue colocado porque en varias secuencias animadas se observó el comportamiento de que los objetos desaparecen después de terminar su rotación.

El funcionamiento es muy similar al de la función anterior; de la misma forma devuelve una promesa, lo que garantiza un control preciso sobre el estado de la animación, y permite que las operaciones finalicen correctamente antes de proceder a la siguiente acción. De la misma forma, contiene una lista de objetos que están rotando y una variable que identifica si la animación está en curso, para evitar rotación infinita o comportamientos erráticos.

La flexibilidad de la función está sustentada por los siguientes parámetros que permiten personalizar las características de la animación:

- `nombreModelo`: `ID de Studio` del objeto sobre el que se aplicará la rotación.
- `eje`: Eje de rotación (por ejemplo, `rx`, `ry`, `rz`).
- `vueltasTotales`: Número total de vueltas a realizar en la rotación.
- `visibility`: Controla si el modelo es visible o no tras completar la rotación.
- `timingInterval`: Intervalo de tiempo en milisegundos entre cada incremento de rotación.

Debido a que ya se explicó la lógica de la animación de traslación, y al ser muy similar la seguida por la función de rotación, el funcionamiento de esta se numerará en la siguiente lista simplificada de pasos:

1. La función `Rotación` recibe cinco parámetros: el nombre del modelo, el eje de rotación, el número total de vueltas, la visibilidad del modelo al final de la animación y el intervalo de tiempo.
2. La promesa controla el proceso, resolviéndose cuando la animación finaliza satisfactoriamente o rechazándose si se interrumpe.
3. El ángulo del modelo se inicializa en 0, y si el modelo no está en su posición original, se restablece a esta.
4. A través de un ciclo, la rotación avanza en incrementos de 45 grados hasta que se completa el número de vueltas especificado.
5. Si se presiona el botón de nuevo, la rotación se cancela y se remueve el modelo de la lista de rotación.
6. La velocidad del movimiento se ajusta mediante el intervalo de tiempo definido.
7. Al finalizar, si la visibilidad está habilitada, el modelo se oculta y se resuelve la promesa, indicando que la rotación fue exitosa.

La versión sintetizada del código de la función rotación, manteniendo los principios fundamentales de su funcionamiento, se muestra en el Código 3.14.

```
$scope.Rotacion = function(nombreModelo, eje, vueltasTotales, visibility) {
  return new Promise((resolve, reject) => {                              // Retorna una promesa
    var rInicial = 0;                                                    // Inicializa angulo a 0
    $scope.view.wdg[nombreModelo]['visible'] = true;                     // Hacer visible el modelo
    if ($scope.view.wdg[nombreModelo][eje] != rInicial) {                // Verificar posicion inicial
      $scope.view.wdg[nombreModelo][eje] = 0;                            // Restablecer a 0
    }
    var updateAngulo = function() {                                      // Funcion para actualizar angulo
    if ($scope.view.wdg[nombreModelo][eje] == 0 && listaObjetosRotacion.includes(nombreModelo)) {
        listaObjetosRotacion.splice(listaObjetosRotacion.indexOf(nombreModelo), 1); // Quitar nombre
        reject('Se presiono el boton de nuevo');                         // Rechazar promesa
        return false;                                                    // Terminar funcion
      }
      if (!listaObjetosRotacion.includes(nombreModelo)) {                // Verificar si no esta en la lista
        listaObjetosRotacion.push(nombreModelo);                         // Agregar nombre
      }
      if ($scope.view.wdg[nombreModelo][eje] < vueltasTotales * 45) {    // Verificar angulo total
        $scope.view.wdg[nombreModelo][eje] += 45;                        // Aumentar angulo
        $scope.$apply();                                                 // Actualizar pantalla
        $timeout(updateAngulo, timingInterval);                          // Llamar nuevamente con intervalo
      } else {
        if (visibility == true) {
          $scope.view.wdg[nombreModelo]['visible'] = false;              // Hacer invisible al finalizar
        }
        listaObjetosRotacion.splice(listaObjetosRotacion.indexOf(nombreModelo), 1); // Quitar nombre
        resolve('Operacion de rotacion exitosa');                        // Resolver promesa
      }
    };
    updateAngulo();                                                      // Iniciar el ciclo
  });
};
```

Código 3.13 Función simplificada para realizar animación de rotación.

3.4.3. Parpadeo

El parpadeo consiste en la alternancia cíclica del color de un *model item*, entre un color distintivo y el color original de la pieza. Esto proporciona una indicación visual clara y dinámica en la experiencia de AR, permitiendo que una pieza destaque o llame la atención del usuario.

Este tipo de animación se utiliza comúnmente en contextos donde se desea identificar una pieza o proceso rápidamente. Por ejemplo, en un proyecto de prueba de Vuforia Studio para mostrar el reemplazo de una batería de un cuadricóptero, el tornillo que libera la tapa parpadea en rojo cuando es momento de quitarlo.

Los parámetros necesarios para usar esta animación son:

- `nombreModelo`: `ID de Studio` del objeto sobre el que se aplicará el parpadeo.
- `blinding`: Define la cantidad de veces que se alternará el color del modelo y controla la duración del efecto de parpadeo.

El funcionamiento del parpadeo, detallado en el Código 3.14, comienza con la inicialización de `contadorcolor` en 0 y la actualización de las variables de estado `enEjecucion` y `enEjecucionAux`. La lógica principal reside en la función `updateColor`, la cual se ejecuta de forma asíncrona mediante `$timeout`. Esta función valida primero la continuidad del proceso; si el estado no ha variado, compara el `contadorcolor` con el umbral `blinding`. Mientras el contador sea inferior a dicho umbral, el sistema alterna el color del modelo cada cinco ciclos, forzando la actualización de la `vista` con `$scope.apply()`. Una vez que el contador iguala a `blinding`, el sistema restaura el color original del modelo y reinicia el ciclo, garantizando así un efecto visual constante y sincronizado.

```
$scope.Parpadeo = function(nombreModelo, blinding) {
  var contadorcolor = 0;                                  // Contador para frecuencia de parpadeo
  var enEjecucionAux = enEjecucion;                       // Estado anterior de ejecucion
  enEjecucion = !enEjecucion;                             // Cambia el estado de ejecucion
  var updateColor = function() {                          // Funcion para modificar el color
    if (enEjecucionAux == enEjecucion) return false;      // Termina si el estado cambio
    if (contadorcolor < blinding) {
      if (contadorcolor % 5 == 0) {                       // Cambia color cada 5 ciclos
        $scope.view.wdg[nombreModelo]['color'] =
          ($scope.view.wdg[nombreModelo]['color'] == 'rgb(255,125,0)')
          ? 'None'                                        // Si la condicion es verdadera color = none
          : 'rgb(255,125,0)';                             // Si la condicion es falsa, color = naranja
      }
      $scope.$apply();                                    // Actualiza la vista
      contadorcolor++;                                    // Incrementa el contador
      $timeout(updateColor, timingInterval);              // Llama a la funcion nuevamente
    } else {
      contadorcolor = 0;                                  // Resetea el contador
      $scope.view.wdg[nombreModelo]['color'] = 'None'; // Vuelve al color original
    }
  };
  updateColor();                                          // Inicia el ciclo
};
```

Código 3.14 Función simplificada para realizar animación de parpadeo.

3.4.4. Destacar elemento

Dentro de la `vista Conocer planta`, se cuenta con la función `$scope.destacarInstrumento`, que se encarga de resaltar un *model item* específico, seleccionado por el usuario al tocarlo con su dedo. Cuando esto sucede, el elemento se muestra de manera más prominente, con un cambio de color suave, gradual y cíclico, y se atenúan todos los demás equipos del sistema de manejo de líquidos.

La función `destacarInstrumento()`, cuyo código se muestra en 3.15, se activa inicialmente verificando si la variable `mostrarPopup` es verdadera; si esta condición se cumple, se ejecuta el resto de la función.

Para reiniciar el estado, se llama a `$scope.default()`, lo cual cancela al intervalo (`$interval`) de `parpadeoInstrumento`, establece la opacidad de todos los equipos a 1 y desaparecen los *model items* auxiliares.

Posteriormente, la función ajusta la opacidad de los instrumentos iterando sobre `listaInstrumentos`, un objeto que contiene los nombres de todos los componentes disponibles. Si el elemento actual coincide con `nombreInstrumento`, dicho modelo permanece opaco, mientras que el resto del sistema reduce su opacidad a 0.5. En el caso de los rotámetros, el código verifica si el objeto seleccionado es `RotametroF` o `RotametroR` para establecer la opacidad de sus respectivas placas de nivel (`placaNivelF` o `placaNivelR`) en 1. Esta operación es necesaria debido a que el modelo completo de cada rotámetro está compuesto por dos `model items` independientes que deben sincronizarse.

Luego, se inicia el proceso de parpadeo del instrumento, inicializando la variable `opacity` en 0 y `sentido` en `+`, lo que permite controlar el aumento en el brillo del componente. Se crea una variable denominada `instrumento`, la cual contiene el nombre del objeto con el sufijo 'Aux' para hacer referencia al *model item* auxiliar seleccionado; su visibilidad se establece en `true` para mostrarlo en la interfaz. Asimismo, se activa la variable `parpadeoInst` para indicar que el proceso está en curso.

Finalmente, se utiliza `$interval` para ejecutar un ciclo cada 10 milisegundos que ajusta la opacidad del instrumento auxiliar. En este ciclo, si `mostrarPopup` es verdadero, se obtiene la opacidad actual: si el sentido es '+', la opacidad se incrementa en 0.01 hasta alcanzar 1, momento en que cambia a '-'. Por

el contrario, si el sentido es '-', la opacidad disminuye en 0.01 hasta llegar a 0, regresando entonces a '+'. Si en cualquier momento `mostrarPopup` cambia a falso, se cancela el intervalo de parpadeo.

Para que esta animación funcione correctamente, es necesario posicionar un modelo idéntico al original en la ubicación exacta. Este objeto, denominado instrumento auxiliar, debe configurarse con la propiedad `Siempre arriba`, opacidad inicial en 0 y un color distintivo. De esta forma, al tocar un instrumento, el modelo auxiliar se superpone y la variación de su opacidad genera la ilusión de un resplandor variable y cíclico.

```
$scope.destacarInstrumento = function(nombreInstrumento){          // Inicia la funcion con el ID del Model Item
  if(mostrarPopup){                                                // Verifica si el popup debe mostrarse
    $scope.default();                                              // Restablece el estado de la vista
    for(var i = 0; i < listaInstrumentos.length; i++){             // Itera sobre todos los instrumentos
      if(listaInstrumentos[i] == nombreInstrumento){               // Si el instrumento coincide con el selected
      }else{                                                       // (se omite opacidad de placas de nivel)
        $scope.view.wdg[listaInstrumentos[i]]['opacity'] = 0.5;    // Para los demas, reduce la opacidad a 0.5
      }
    }
    var opacity = 0;                                               // Inicializa la opacidad en 0
    var sentido = '+';                                             // Establece la direccion del parpadeo
    var instrumento = nombreInstrumento + 'Aux';                   // Crea el nombre del modelo auxiliar
    $scope.view.wdg[instrumento]['visible'] = true;                // Hace visible el modelo auxiliar
    parpadeoInst = true;                                           // Indica que el parpadeo esta activo
    parpadeoInstrumento = $interval(function(){                    // Inicia el intervalo de parpadeo
      if(mostrarPopup){                                            // Verifica si el popup sigue visible
        opacity = $scope.view.wdg[instrumento]['opacity'];         // Obtiene la opacidad actual
        if(sentido == '+'){                                        // Si el sentido es hacia arriba
          if(opacity < 1){                                         // Si la opacidad es menor a 1
            $scope.view.wdg[instrumento]['opacity'] = opacity + 0.01;  // Aumenta la opacidad
          }else{                                                   // Si ha alcanzado 1
            sentido = '-';                                         // Cambia el sentido a abajo
          }
        }else{                                                     // Si el sentido es hacia abajo
          if(opacity > 0){                                         // Si la opacidad es mayor a 0
            $scope.view.wdg[instrumento]['opacity'] = opacity - 0.01;  // Disminuye la opacidad
          }else{                                                   // Si ha alcanzado 0
            sentido = '+';                                         // Cambia el sentido a arriba
          }
        }
      }else{                                                       // Si el popup no se debe mostrar
        $interval.cancel(parpadeoInstrumento);                     // Cancela el intervalo de parpadeo
      }
    }, 10);                                                        // Intervalo de 10 milisegundos
  }
};
```

Código 3.15 Función para realizar animación de brillo.

3.4.5. Animación de flechas

La función `animacionAgua` controla el movimiento de las flechas que representan el flujo en los circuitos de transferencia y recirculación del sistema de tanques, tal como se muestra en la Figura 3.13. A continuación, se describen sus componentes y lógica de operación.

En primer lugar, se establecen variables clave: `timingInterval` define el intervalo de actualización en 10 milisegundos; `coordenadaOriginalFlechaAgua` guarda la posición inicial de la primera flecha (`flechaAgua-1`), mientras que `distancia` calcula su desplazamiento. Asimismo, se emplea `promesa` para almacenar el resultado del proceso de traslación.

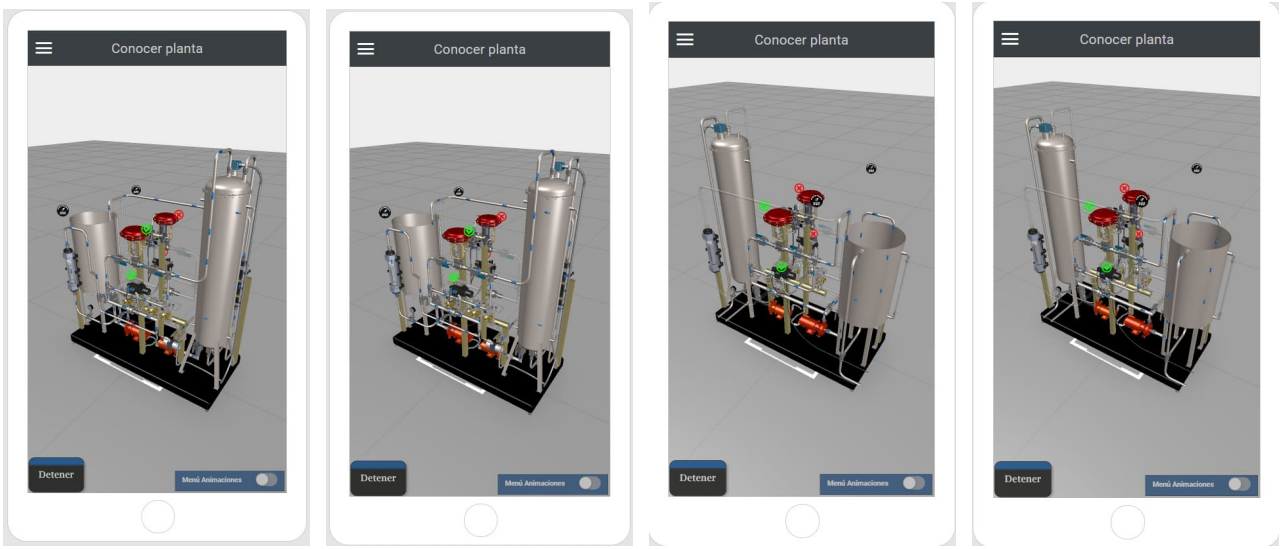
La ejecución principal reside en una función interna denominada `iniciarAnimacion`; esta utiliza un bucle `$interval`, llamado `traslacionFlechas`, encargado de actualizar la posición de los elementos. El proceso comienza invocando a `verificarEstados()` (Código 3.10); si `estadoAnimacion` ya es `true`, la función retorna `false` para evitar ejecuciones concurrentes o comportamientos inesperados; de lo contrario, el estado cambia a activo.

Después, se inicia la traslación de `flechaAgua-1` en el eje `y` con un avance de `0.3`, generando una promesa. En este punto se activa el bucle `traslacionFlechas`, que recorre desde la flecha 2 hasta la 55. En cada iteración, el sistema verifica si se ha presionado el botón de detención para cancelar el intervalo si es necesario; de lo contrario, determina la dirección y el eje de movimiento según las propiedades de cada flecha.

Para cada elemento del grupo, se calcula el desplazamiento relativo a la primera flecha y se actualizan sus posiciones en función de dicha distancia y dirección. Al finalizar el recorrido, si la flecha inicial ha alcanzado su destino, se restablece `estadoAnimacion` a `false` y se termina el bucle.

Finalmente, se evalúa la promesa de traslación: si se cumple, la animación se reinicia llamando nuevamente a `iniciarAnimacion()`; si falla, se cancela el intervalo y se registra el error en la consola. Cabe mencionar que `animacionAgua` se ejecuta inicialmente tras un retraso de 50 milisegundos mediante `$timeout`.

Este enfoque, basado en el avance de la primera flecha para definir el desplazamiento de las demás, optimiza significativamente los recursos de procesamiento. Debido a la extensión del código fuente, este se ha incluido en los apéndices de la tesis (ver Código A.2).



(a) Estado 1: flechas de transferencia. (b) Estado 2: flechas de transferencia. (c) Estado 1: flechas de recirculación. (d) Estado 2: flechas de recirculación.

Figura 3.13 Resultados de la función `animacionAgua()`. [12]

3.5. Seguimiento de objetos

Cada tipo de *target* en Vuforia está diseñado para situaciones específicas, por lo que la selección de un método de seguimiento adecuado en una experiencia desarrollada en Vuforia Studio es crucial, pues determina la precisión, estabilidad y el contexto de interacción entre el usuario, el contenido digital y el entorno físico.

Para garantizar un correcto funcionamiento del seguimiento, se siguieron las recomendaciones de [Pitser, 2021], las cuales señalan que, al emplear marcadores, estos deben estar correctamente

[12] Figura de autoría propia.

localizados y poseer un tamaño adecuado; asimismo, el entorno requiere condiciones favorables de iluminación, mínimas vibraciones y, en caso de existir desplazamientos, que estos sean lentos. Siguiendo esto, se utilizan los métodos nombrados a continuación con su respectiva filosofía de diseño.

3.5.1. *Spatial Target*

Este objetivo permite colocar contenido sobre superficies horizontales sin necesidad de marcadores, es decir, el seguimiento se realiza en función del entorno, no de un objeto específico. Además, incluye de manera predeterminada la capacidad de mover, rotar y escalar modelos.

Es ideal para revisiones y visualización de diseños de objetos o productos, sin necesidad de tenerlos físicamente; también es útil para pruebas de uso en entornos físicos reales. Para lograr resultados óptimos, es necesario que la iluminación sea estable y moderada, sin deslumbramientos, y que la superficie sea adecuada [Pitser, 2021].

En la aplicación desarrollada, el *spatial target* se utiliza para visualizar la planta de tratamiento de líquidos completa sobre una superficie, junto con las superposiciones digitales asociadas a ese modelo, lo que permite visualizarlo de forma accesible en casi cualquier entorno. Además, este objetivo se usó para mostrar animaciones del funcionamiento de la instrumentación específica de la planta, ofreciendo capacidades de visualización imposibles en la vida real, como ver el interior de los equipos y observar las interacciones de piezas internas que realiza un actuador al ejecutar su función principal.

3.5.2. *Image Target*

Los *Image Targets* se usan de manera similar al objetivo anterior, pero la principal diferencia radica en el objeto de seguimiento; por ello, este método es ideal en situaciones donde hay una imagen impresa disponible en el entorno y se necesita un seguimiento rápido y sencillo, sin depender de una superficie. Es crucial asegurarse de que la imagen tenga buena calidad y contraste para un reconocimiento eficiente.

En la experiencia creada, el usuario puede elegir entre *spatial target* o *image target* en determinadas `vistas`, lo que aporta flexibilidad a la aplicación al permitirle visualizar el contenido de AR en distintas circunstancias y contextos.

3.5.3. *Model Target*

Los *model targets* se recomiendan para mostrar instrucciones complejas de tareas, procedimientos de servicio, operación y mantenimiento de equipos; especialmente en escenarios donde el uso de *ThingMarks* no resulta viable. Por este motivo, se implementó este tipo de seguimiento en diversas `vistas` del sistema.

Esta técnica es fundamental en aplicaciones industriales, de capacitación o mantenimiento, donde se requiere superponer información digital directamente sobre maquinaria física. Su elección garantiza una experiencia altamente interactiva y precisa, ya que mantiene su funcionalidad bajo diversas condiciones de iluminación y proximidad, siempre que el equipo sea claramente distinguible y los modelos CAD posean exactitud en su geometría y dimensiones.

3.6. Despliegue y publicación de la experiencia

El proceso de publicación en Vuforia Studio comienza con la creación de experiencias de AR, que pueden contener elementos 2D, 3D y datos dinámicos provenientes de ThingWorx. Una vez publicadas, las experiencias son accesibles escaneando un código QR, enlaces o *ThingMarks* con la aplicación Vuforia View. Para gestionar estas experiencias, Vuforia Studio utiliza un servidor seguro y escalable llamado **Experience Service**, el cual hospeda las experiencias y las hace accesibles desde cualquier dispositivo compatible.

La publicación de las experiencias se realizó en un entorno de red local cerrado y sin acceso a internet, lo que permitió alojar el servidor de visualización en una máquina virtual independiente de servicios externos. El proceso en Vuforia Studio consistió en vincular el proyecto a la dirección del servidor local, configurar el acceso público para habilitar la descarga de contenido fuera de línea y definir la `vista` inicial.

Una vez finalizada la configuración y ejecutada la publicación desde la interfaz (Figura 2.5), el sistema genera un código QR o un enlace de acceso, disponibles en la pestaña de compartir. Estos permiten cargar la experiencia en Vuforia View, ya sea mediante la red local para comunicarse con el servidor o de forma directa en el dispositivo si el contenido fue descargado previamente para su uso fuera de línea.

Capítulo 4

Evaluación

Como se ha expuesto, la experiencia de AR desarrollada tiene como propósito asistir en tareas operativas de sistemas productivos, específicamente en mantenimiento, operación, supervisión y capacitación, además de impulsar la integración de herramientas tecnológicas en el sector industrial. Para lograrlo, aspectos como el rendimiento, la fiabilidad y la experiencia del usuario resultan cruciales para garantizar la usabilidad y maximizar la relevancia de la solución propuesta [Pachhandara, 2022].

Por ello, la evaluación del presente trabajo consiste en analizar el rendimiento computacional de la experiencia, con el fin de identificar y corregir posibles brechas de desempeño antes de implementar la aplicación en el entorno industrial descrito previamente.

4.1. Evaluación técnica y funcional

La solución desarrollada en Vuforia Studio integra modelos CAD y permite la interacción en tiempo real, por lo que el rendimiento técnico impacta directamente en la experiencia del usuario. Con el fin de asegurar la eficiencia de la aplicación, la presente evaluación se centró en tres aspectos clave: tiempos de respuesta, uso de recursos computacionales y fidelidad visual de los elementos. Dichas dimensiones se cuantificaron a través de las métricas detalladas en la sección 4.1.

Para llevar a cabo las evaluaciones se usaron tres dispositivos: una computadora personal (PC) que ejecutó Vuforia Studio en Windows y dos teléfonos móviles con sistema operativo Android. Los datos se obtuvieron de la forma detallada en el siguiente listado:

- **Entorno de escritorio (PC):** Las pruebas se realizaron en el navegador Microsoft Edge, plataforma donde se ejecuta el entorno de Vuforia y las vistas previas de la aplicación. Los datos se obtuvieron mediante las herramientas de desarrollo (DevTools), empleando funciones como *Performance* y *Lighthouse* para analizar los tiempos de respuesta. Por su parte, se utilizó el `Administrador de tareas de Windows` para monitorear el consumo de recursos computacionales.
- **Plataformas móviles:** Se empleó el *software* Apptim para Windows, una herramienta que permite evaluar el rendimiento de aplicaciones nativas en sistemas Android. Esta solución cuenta con la capacidad de ejecutar pruebas automatizadas para analizar tiempos de renderizado, consumo de energía y uso de recursos de procesamiento, además de capturar errores y excepciones. Finalmente, el sistema genera un informe detallado con la métrica recolectada para su posterior análisis [Gavilanes, 2020].

Todas las pruebas se realizaron ejecutando la `vista Conocer planta` en las modalidades de `Operación` y `Capacitación`. Se seleccionaron estas opciones debido a que sus *overlays* integran la mayor cantidad de animaciones simultáneas e interacciones con modelos y botones; por lo tanto, representan la `vista` con mayor carga de procesamiento, tanto a nivel gráfico como de gestión de datos.

Métricas de rendimiento

Las métricas clave para la evaluación técnica se detallan en la Tabla 4.1, donde se destacan la tasa de fotogramas por segundo (FPS), la latencia de interacción, el consumo de batería y el uso de memoria RAM. Asimismo, se analiza el aprovechamiento de la unidad central de procesamiento (CPU) y de la unidad de procesamiento gráfico (GPU), junto con la calidad del seguimiento. Para cada indicador se especifica su definición, objetivo y valor esperado, además de las herramientas empleadas para su medición en plataformas Android y Windows.

Métrica	Definición	Objetivo	Métrica Deseada	Herramientas
Tasa de fotogramas por segundo (FPS)	Número de fotogramas renderizados por segundo	Mantener una experiencia fluida	Mínimo 30 FPS (ideal 60 FPS)	- Android: Apptim - PC: DevTools de Edge
Latencia de interacción	Tiempo entre acción del usuario y respuesta visual	Minimizar la latencia	Menos de 20 ms	- Android: Apptim - PC: DevTools de Edge
Consumo de batería	Energía consumida por la aplicación	Reducir el consumo de batería	mAh por minuto de uso	- Android: Apptim - PC: No aplica
Uso de memoria (RAM)	Cantidad de memoria RAM utilizada por la aplicación	Optimizar el uso de memoria	Menos del 50 % de la RAM disponible	- Android: Apptim - PC: DevTools de Edge
Uso de recursos de procesamiento	Porcentaje de recursos de CPU y GPU que consume	Optimizar la utilización de recursos	Menos del 60 %	- Android: Apptim - PC: DevTools de Edge
Calidad de seguimiento AR	Precisión y estabilidad del rastreo del entorno o de objetos	Mantener seguimiento preciso y estable	Sin interrupciones visibles (> 1s)	- Android: Vuforia View - PC: No aplica

Tabla 4.1 Métricas de evaluación técnica y herramientas de medición.

Respecto a algunas métricas de desempeño adicionales, el SDK utilizado por Vuforia Studio garantizó las siguientes funcionalidades:

- **Resolución y fidelidad visual:** Las experiencias exhiben una alta calidad gráfica en todos los dispositivos compatibles con Vuforia View, condicionada a que tanto los modelos CAD como los elementos visuales posean una resolución adecuada.
- **Consumo de recursos de red:** Las aplicaciones permiten su ejecución local tras la descarga inicial, lo que restringe el consumo de datos exclusivamente a tareas de actualización o transferencia de nuevos contenidos.
- **Accesibilidad:** Vuforia View es compatible con una amplia gama de dispositivos Android e iOS, siempre que estos soporten las librerías ARCore o ARKit, respectivamente; los detalles específicos sobre esta compatibilidad se profundizan en el análisis de resultados.

A continuación se presentan las evaluaciones, organizadas en subsecciones para cada dispositivo, donde se describe brevemente la metodología aplicada y se enumeran los resultados obtenidos.

4.1.1. Pruebas en computadora personal

La PC en la que se realizaron las pruebas cuenta con una GPU Intel(R) UHD Graphics; en cuanto al CPU se trata de un Intel(R) Core(TM) i5-10500H CPU @ 2.50GHz acompañado de 16 GB de memoria RAM.

+En cuanto a la metodología, se recopilaron datos de rendimiento en dos escenarios: los *overlays* de `Capacitación` y `Operación`. Los escenarios fueron evaluados mediante dos herramientas: DevTools para medir la tasa de fotogramas por segundo (FPS), la latencia y el consumo de memoria RAM; y el `Administrador de tareas de Windows` para obtener el uso de recursos de procesamiento, específicamente de la GPU y el CPU.

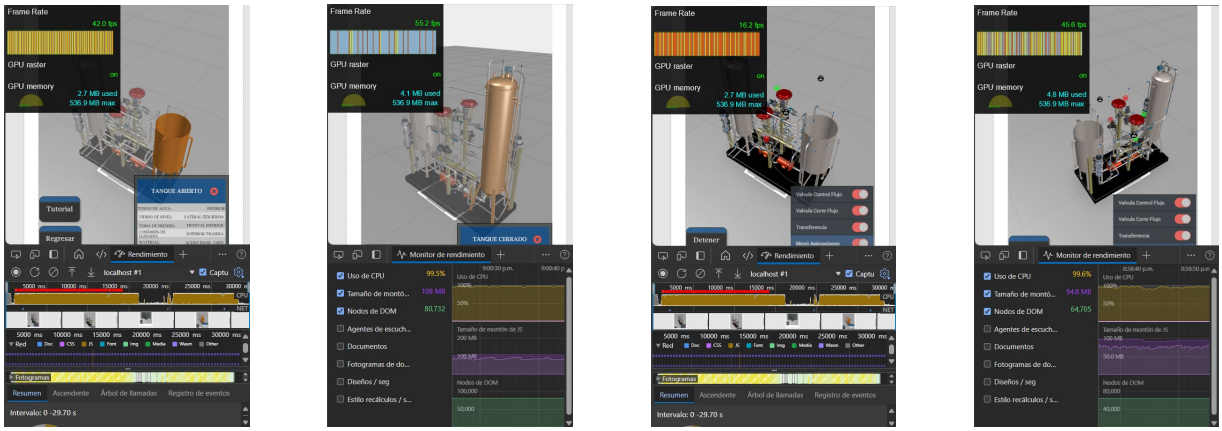
FPS, latencia y RAM

Se empleó la sección Performance de DevTools para realizar dos grabaciones comparativas: una bajo condiciones normales y otra aplicando una restricción de procesamiento. En la segunda grabación se redujo la capacidad de cómputo del PC al 25 % de su rendimiento nominal, con el fin de analizar el rendimiento con menores recursos. En ambos escenarios, se utilizó un visualizador integrado para monitorear en tiempo real la tasa de FPS.

Los resultados obtenidos se organizaron en la Tabla 4.2, la cual detalla los FPS promedio, la latencia de respuesta de los botones y el intervalo transcurrido hasta el comienzo de la animación del flujo de agua (`Inicio animación`). Asimismo, se incluyen referencias a las capturas de pantalla de DevTools que validan los datos presentados.

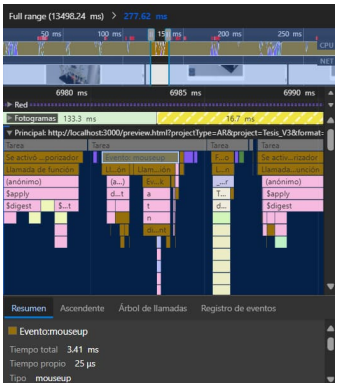
Escenario	Limitación de procesador	FPS Promedio	Latencia (ms)	Inicio animación (ms)	Consumo RAM (MB)
Capacitación	Sí	42.0 (fig: 4.1a)	3.41 (fig: 4.2a)	N/A	645
	No	55.2 (fig: 4.1b)	1.96 (fig: 4.2b)	N/A	645
Operación	Sí	14.4 (fig: 4.1c)	6.83 (fig: 4.2c)	350 (fig: 4.3a)	645
	No	47.4 (fig: 4.1d)	1.15 (fig: 4.2d)	192 (fig: 4.3b)	645

Tabla 4.2 Resultados de tasa de FPS y latencia en diferentes escenarios.

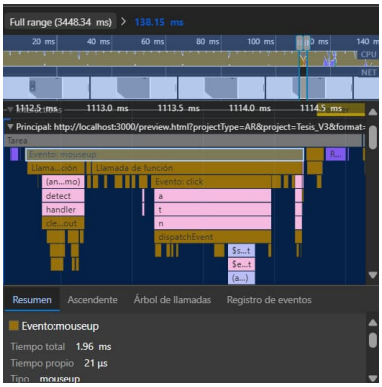


(a) *Overlay* de capacitación con rendimiento limitado. (b) *Overlay* de capacitación con rendimiento normal. (c) *Overlay* de operación con rendimiento limitado. (d) *Overlay* de operación con rendimiento normal.

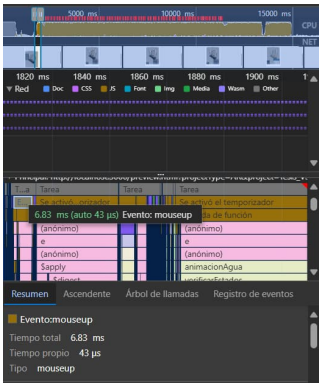
Figura 4.1 Resultados de la evaluación de FPS en la vista `Conocer planta`. [1]



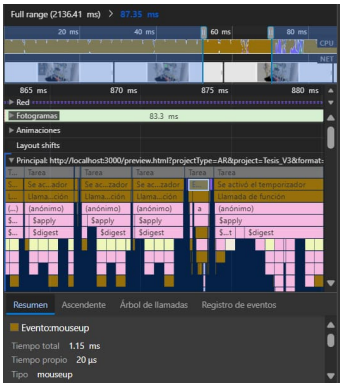
(a) Capacitación con limitación de procesador.



(b) Capacitación sin limitación de procesador.

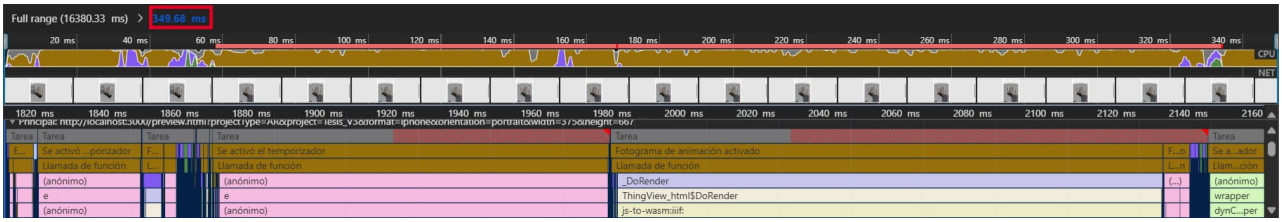


(c) Operación sin limitación de procesador.

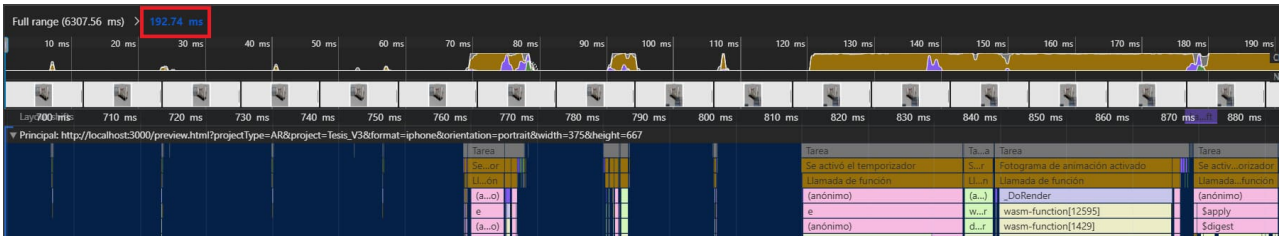


(d) Operación sin limitación de procesador.

Figura 4.2 Latencia de los botones de navegación en la vista `Conocer planta`. [2]



(a) Latencia con limitación de procesador.



(b) Latencia sin limitación de procesador.

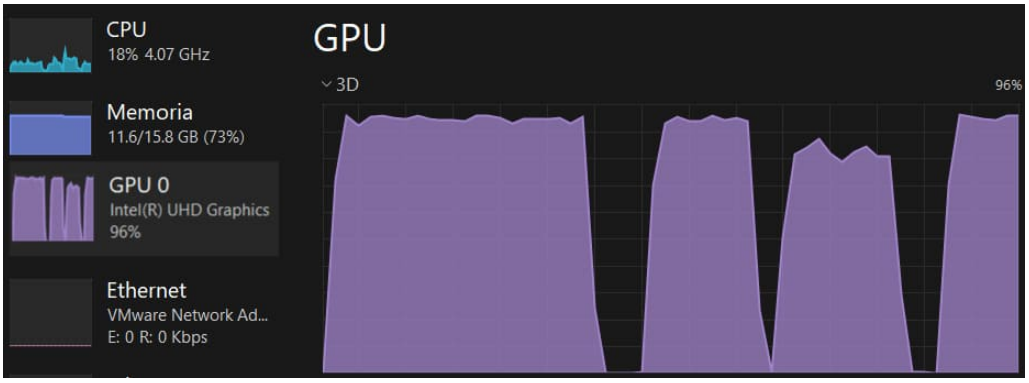
Figura 4.3 Latencia de los botones de animación en la vista `Conocer planta`. [3]

[1]Figuras de autoría propia.
[2]Figuras de autoría propia.
[3]Figuras de autoría propia.

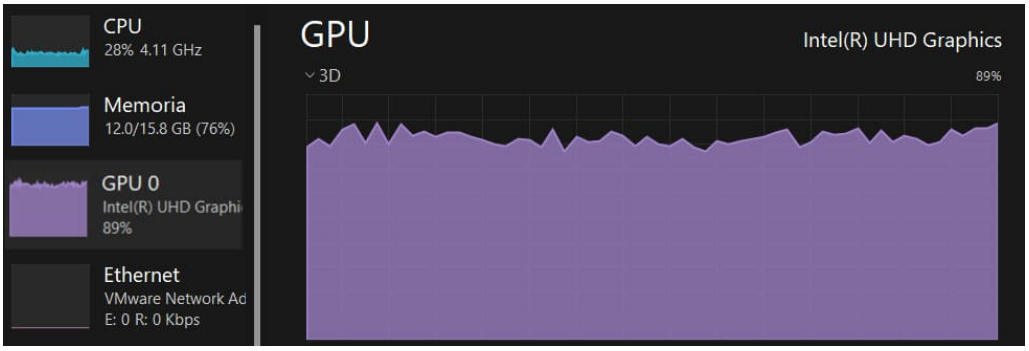
La medición del consumo de memoria RAM se obtuvo utilizando las DevTools en la sección de `memoria`. Por su parte, los datos del uso de recursos de procesamiento fueron recopilados a través del `Administrador de Tareas de Windows`, mientras se ejecutaban los dos *overlays* mencionados anteriormente; estos resultados se muestran en la Tabla 4.3.

Escenario	**Uso de GPU (%)**	**Uso de CPU (%)**
`Capacitación`	96 (fig: 4.4a)	18 (fig: 4.4a)
`Operación`	89 (fig: 4.4b)	28 (fig: 4.4b)

Tabla 4.3 Consumo de memoria RAM y GPU en PC en diferentes escenarios.



(a) Consumos del *overlay* `Capacitación`.



(b) Consumos del *overlay* `Operación`.

Figura 4.4 Consumo de recursos de procesamiento (CPU y GPU) de la vista `Conocer planta`. [4]

4.1.2. Pruebas en dispositivo Android con rendimiento moderado

Para realizar la prueba automatizada en la experiencia de tanques atmosféricos, fue necesario activar los permisos de depuración *USB* del dispositivo utilizado; en este caso fue un equipo modelo Motorola Edge 30, con 8 GB de RAM y 8 núcleos de procesamiento. Una vez conectado el dispositivo al puerto *USB* de la PC y realizada la configuración necesaria dentro de Apptim, se ejecutó una prueba de 4 minutos con 10 segundos, lo que proporcionó los siguientes datos:

[4]Figuras de autoría propia.

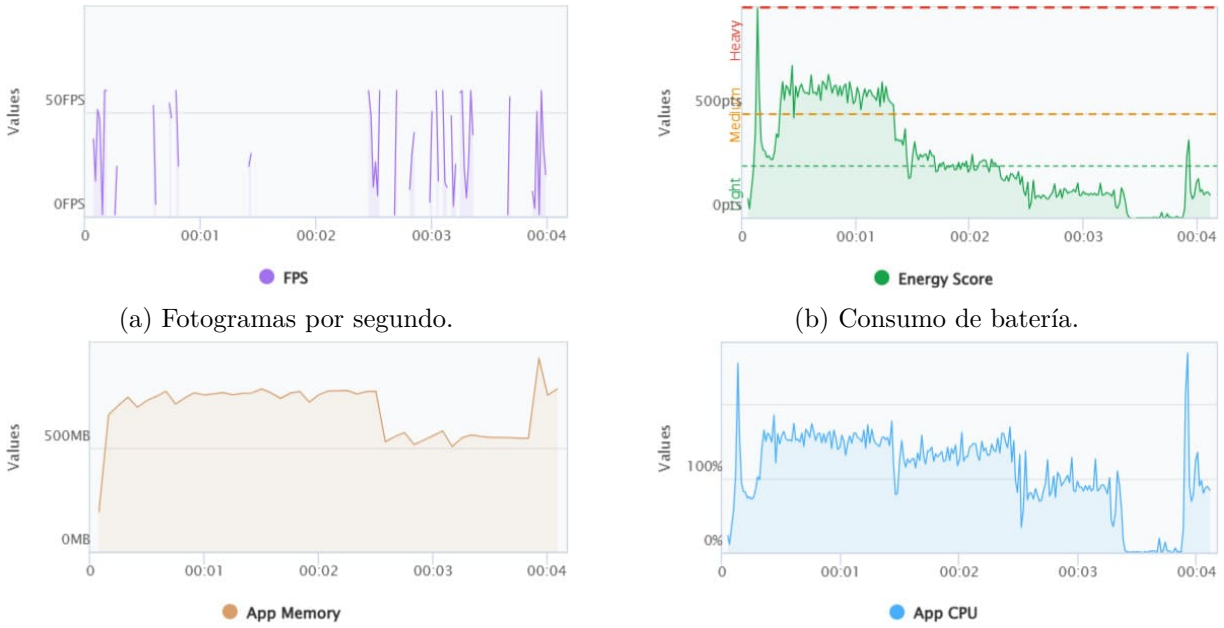
1. **FPS:** La tasa de fotogramas por segundo alcanzó un máximo de 90, un mínimo de 1 y un promedio de 37.2. El mínimo valor de FPS probablemente ocurrió durante el renderizado del modelo (ver Figura 4.5a).

2. **Latencia de interacción:** La latencia máxima registrada fue de 11.1 ms.

3. **Consumo de batería:** Aunque el *software* no proporciona el consumo de batería en miliamperios por hora (mAh), estima el gasto energético del CPU y asigna una puntuación en una escala de 0 a 1000. La experiencia obtuvo una puntuación promedio de 283.2, lo que indica un consumo de batería moderado según esta clasificación (ver Figura 4.5b).

4. **Uso de memoria RAM:** La aplicación utilizó un máximo de 903.54 MB, un mínimo de 186.63 MB y un promedio de 656.74 MB. Esto representa aproximadamente el 8.02 % del total de 8 GB de RAM disponibles en el dispositivo (ver Figura 4.5c).

5. **Uso de CPU:** El uso de la CPU mostró un máximo del 269.52 %, un mínimo de 0.96 % y un promedio de 110.46 %. Dado que el dispositivo tiene 8 núcleos con un uso máximo del 800 %, el consumo promedio representa aproximadamente el 13.81 % de la capacidad total (ver Figura 4.5d).



(a) Fotogramas por segundo.

(b) Consumo de batería.

(c) Uso de memoria RAM.

(d) Uso de CPU.

Figura 4.5 Resultados de la evaluación técnica en un dispositivo de rendimiento moderado. [5]

4.1.3. Pruebas en dispositivo Android de rendimiento limitado

Para garantizar un rendimiento adecuado en dispositivos de distintas gamas utilizados por los ingenieros, se realizó una prueba de rendimiento en un equipo de menor capacidad, un Samsung Galaxy A54, con 8 núcleos de procesamiento y 8 GB de RAM. Al lograr un buen desempeño en este dispositivo menos

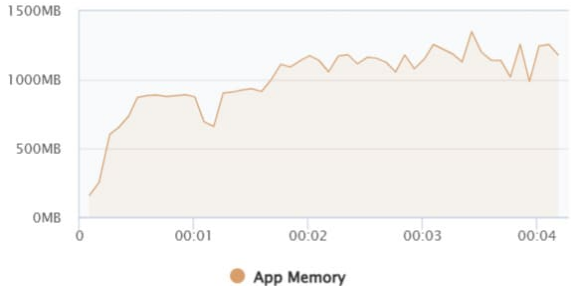
[5] Figuras de autoría propia.

potente, se puede inferir que la aplicación ofrecerá un rendimiento óptimo en otros dispositivos más avanzados.

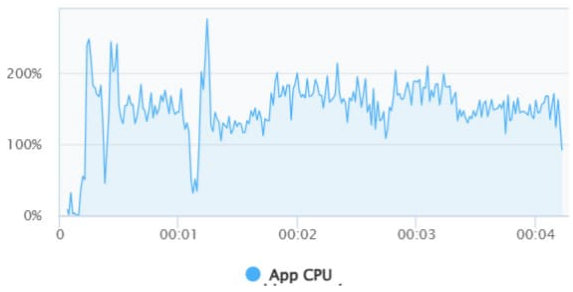
1. **FPS:** La tasa de fotogramas por segundo alcanzó un máximo de 60, un mínimo de 1 y un promedio de 22.34 (ver Figura 4.6a).
2. **Latencia de interacción:** La latencia máxima registrada fue de 16.67 ms.
3. **Consumo de batería:** El nivel máximo de consumo de batería fue de 998.98 pts, con un mínimo de 0.15 y un promedio de 461.11 pts, lo que indica un consumo moderado de energía (ver Figura 4.6b).
4. **Uso de memoria RAM:** La aplicación utilizó un máximo de 1318.27 MB de RAM, un mínimo de 154.92 MB y un promedio de 980.04 MB, lo que representa aproximadamente el 11.96 % de los 8 GB disponibles en el dispositivo (ver Figura 4.6c).
5. **Uso de CPU:** El uso de CPU mostró un máximo del 276.04 %, un mínimo de 0.99 % y un promedio del 150.45 %. Dado que el dispositivo tiene 8 núcleos y la capacidad máxima de uso es del 800 %, el consumo promedio representa aproximadamente el 18.81 % del total disponible (ver Figura 4.6d).

(a) Fotogramas por segundo.

(b) Consumo de batería.



(c) Uso de memoria RAM.



(d) Uso de CPU.

Figura 4.6 Resultados de la evaluación técnica en un dispositivo con rendimiento limitado. [6]

4.2. Análisis de resultados

La Tabla 4.4 presenta un resumen de los datos obtenidos, con el objetivo de proporcionar una visión general que permita un análisis del rendimiento de la experiencia bajo diferentes condiciones.

[6] Figuras de autoría propia.

Dispositivo	Escenario	*Limitación de CPU*	*FPS Promedio*	Latencia (ms)	Inicio Anim. (ms)	*Consumo RAM (MB)*	Uso CPU (%)	Uso GPU (%)	Consumo Batería
PC	Capacitación	Sí	42.0	3.41	N/A	645	18	96	N/A
	Capacitación	No	55.2	1.96	N/A	645	18	96	N/A
	Operación	Sí	14.4	6.83	350	645	28	89	N/A
	Operación	No	47.4	1.15	192	645	28	89	N/A
Android Medio	Capacitación /Operación	N/A	37.2	11.1	N/A	656.74	110.46 (13.81 %)	N/A	283.2
Android Bajo	Capacitación /Operación	N/A	22.34	16.67	N/A	980.04	150.45 (18.81 %)	N/A	461.11

Tabla 4.4 Resultados de rendimiento en PC y Android.

Análisis de PC

Los datos obtenidos en la evaluación para PC muestran un rendimiento óptimo bajo condiciones normales de procesamiento, manteniendo una tasa de FPS cercana al valor ideal de 60 FPS. La latencia de interacción registrada es mínima, con valores que oscilan entre 1.15 ms y 1.96 ms, situándose significativamente por debajo del umbral de tolerancia de 20 ms que garantiza una percepción de respuesta instantánea. En cuanto a la memoria RAM, el consumo representó solo el 4.03 % de los 16 GB disponibles, mientras que el uso de CPU se mantuvo en un 18 %; ambos indicadores aseguran una capacidad restante suficiente para procesos del sistema. No obstante, la GPU integrada operó al 96 % de su capacidad, lo que sugiere la conveniencia de emplear procesadores gráficos con mayores prestaciones o habilitar el uso de unidades de procesamiento gráfico externas para aliviar la carga del sistema.

En conjunto, estos resultados evidencian la alta optimización de Vuforia Studio para el desarrollo de experiencias complejas en entornos ágiles y estables. Los datos muestran que el equipo de cómputo destinado a la creación de estas experiencias debe poseer especificaciones de rendimiento similares a las evaluadas, con especial énfasis en la potencia de procesamiento gráfico para garantizar un flujo de trabajo óptimo en aplicaciones de AR.

Análisis de dispositivos Android

La evaluación en dispositivos Android revela áreas con oportunidad de optimización. Si bien la tasa promedio de FPS del dispositivo Edge 30 (37.2 FPS) se sitúa en un rango funcional, el modelo Galaxy A54 opera por debajo del estándar recomendado de 30 FPS. En cuanto a la latencia de interacción, ambos equipos presentan valores superiores a los de la PC; no obstante, se mantienen por debajo del umbral de 20 ms, garantizando que la interactividad y la sensación de respuesta inmediata no se vean comprometidas.

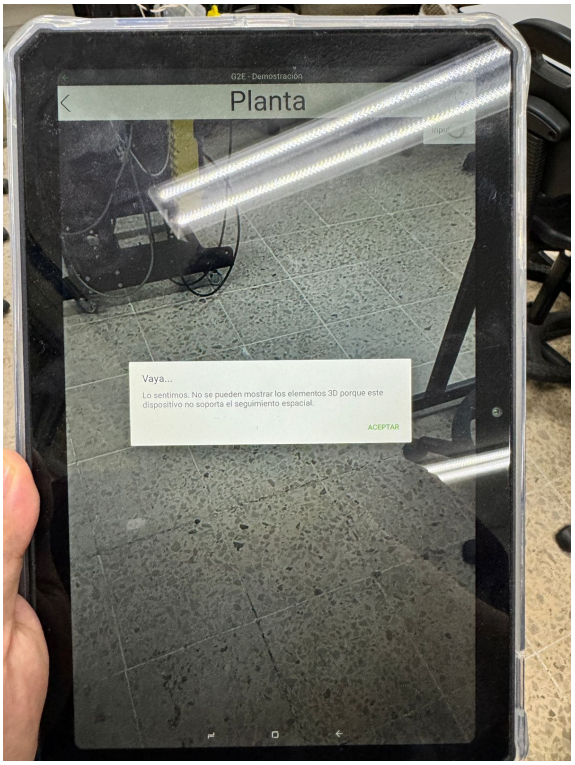
Respecto a la memoria RAM, ambos dispositivos utilizan menos del 50 % de su capacidad. Sin embargo, la brecha del 4 % entre ambos sugiere que existe margen para optimizar la gestión de memoria en *hardware* de prestaciones limitadas. Una tendencia similar se observa en el procesamiento (CPU), donde el consumo se mantiene bajo el 60 %; aun así, los picos de carga registrados reflejan la ejecución de tareas de alta demanda computacional durante la experiencia.

Sobre el consumo energético, el Galaxy A54 registró un índice de 200 unidades superior al Edge 30 según el índice de consumo de la herramienta Apptim, situándose cerca del umbral de alto consumo. Esto evidencia una diferencia significativa en la autonomía al ejecutar aplicaciones de AR en este modelo específico.

Para mitigar estas deficiencias, se recomienda la optimización de estructuras de datos y la simplificación de la complejidad en las animaciones y la UI. Asimismo, es fundamental optimizar los modelos CAD para lograr un equilibrio óptimo entre fidelidad visual y rendimiento gráfico (GPU).

Finalmente, el seguimiento (*tracking*) mostró una estabilidad notable: los modelos conservaron su posición y los gestos de interacción se ejecutaron sin errores. El desempeño del *spatial target* fue óptimo en superficies amplias, opacas y con baja reflexión. Para el *image* y *model target*, la precisión dependió directamente de condiciones de iluminación adecuadas y de la exactitud geométrica del modelo 3D respecto al objeto real. Además, se confirmó que la compatibilidad con los motores ARCore y ARKit es crucial; la ausencia de este requisito impide los métodos de seguimiento de Vuforia, derivando en errores de ejecución como el mostrado en la Figura 4.7.

(a) Dispositivo incompatible con *ARCore*.



(b) Dispositivo incapaz de realizar *tracking*.

Figura 4.7 Alertas mostradas por un dispositivo incompatible con ARCore. [7]

[7] Fotografías de autoría propia.

Capítulo 5

Conclusiones

La experiencia de AR desarrollada en este trabajo fue implementada con éxito, cumpliendo con los requerimientos de diseño y funcionalidad planteados. La estabilidad de las funciones y el rendimiento registrado validan la eficacia de la solución, la cual permite a los ingenieros del Departamento de Ingeniería Eléctrica de la Facultad de Ingeniería de la UNAM visualizar e interactuar activamente con la lógica de operación del sistema de tanques en cualquier dispositivo compatible con Vuforia View y ARCore o ARKit. Así, se confirma el cumplimiento de los objetivos de investigación al entregar una herramienta operativa, accesible y escalable que facilita la supervisión y el mantenimiento mediante el acceso inmediato a recursos visuales y datos técnicos.

Con base en los indicadores de rendimiento y las funcionalidades implementadas, se establece una base técnica que permite trasladar los beneficios de la AR, discutidos en el capítulo 1, hacia la mejora de las actividades operativas reales. El uso de herramientas de grado industrial, como Vuforia Studio, confirma la viabilidad de integrar estas experiencias en procesos con requerimientos técnicos exigentes. Esta implementación no solo refuerza el valor de la AR como una herramienta efectiva dentro de la Industria 4.0, sino que también aporta una referencia práctica para futuras integraciones tecnológicas en entornos de automatización y control.

En complemento, el desarrollo técnico y los criterios de diseño documentados en este trabajo integran una metodología detallada que sirve para entender el desarrollo de aplicaciones de AR mediante *Vuforia Studio*. Esta revisión exhaustiva del proceso de creación puede servir como una guía estructurada para futuros investigadores y desarrolladores interesados en implementar soluciones similares, facilitando la curva de aprendizaje y contribuyendo a la integración efectiva de la tecnología en el sector académico e industrial.

En cuanto al alcance del trabajo, este se centró primordialmente en el diseño, implementación y validación técnica de la aplicación, estableciendo un marco de trabajo para su despliegue en entornos industriales. Por lo tanto, la evaluación se enfocó en métricas de rendimiento y estabilidad del sistema, dejando fuera de este enfoque la cuantificación del impacto en el aprendizaje o la disminución de errores operativos; aspectos que, aunque fundamentales, requieren métodos de evaluación de usuario que trascienden los objetivos de esta implementación.

Asimismo, se identificó como un factor determinante la dependencia de *hardware* de prestaciones específicas, puesto que, para garantizar una ejecución fluida y eficaz del seguimiento, la aplicación requiere dispositivos compatibles con los motores ARCore y ARKit de gama media o alta. Esto representa una restricción para la accesibilidad inmediata de la experiencia en dispositivos de menores

prestaciones, lo que demuestra la importancia de considerar las características de los dispositivos móviles al implementar este tipo de herramientas en el sector industrial.

De forma general, se concluye que la AR posee un potencial significativo para optimizar tareas operativas, más allá del área de capacitación. Las características vistas durante la implementación permiten proyectar aplicaciones en asistencia remota, manuales de operación interactivos y herramientas de supervisión en tiempo real. Estos desarrollos pueden crear beneficios tangibles para la industria, tales como la reducción de tiempos muertos, la mitigación de errores operativos y una mejor adquisición del conocimiento. Dichas ventajas son alcanzables gracias a la flexibilidad y capacidad de personalización que ofrecen los entornos de desarrollo de grado industrial, permitiendo adaptar la visualización de datos y el desarrollo de funcionalidades a las necesidades específicas de cada proceso.

No obstante, se identifica que la integración industrial de esta tecnología enfrenta algunos desafíos, derivados principalmente de los elevados costos de adquisición de *hardware* y de licencias de *software* profesional, los cuales limitan su accesibilidad. Asimismo, los dispositivos de visualización tipo *HMD* (*head-mounted displays*) presentan restricciones de ergonomía y costo, sumado a la ausencia de normas y estándares técnicos que regulen su implementación segura y eficaz en los entornos industriales más exigentes.

A pesar de estos obstáculos, resulta importante continuar en el desarrollo y la documentación de experiencias de AR que generen evidencia sobre sus beneficios operativos en la industria. Fomentar una comunidad técnica especializada y fortalecer la investigación en esta área contribuirá a la maduración tecnológica del sector, facilitando eventualmente una adopción más amplia, accesible y estandarizada en los diversos niveles del entorno industrial.

5.0.1. Trabajo futuro

A partir de los resultados obtenidos y la experiencia técnica adquirida, se observan las siguientes líneas de investigación posibles para continuar con la integración de la AR en el sector industrial y que escapan de los objetivos de esta investigación:

- **Escalabilidad y versatilidad:** Implementar soluciones de AR en sistemas industriales de diversa naturaleza, tales como procesos mecánicos, neumáticos o térmicos, con el fin de evaluar la adaptabilidad del marco de desarrollo propuesto en diversos contextos.
- **Interoperabilidad en la Industria 4.0:** Explorar la integración de la AR con protocolos de comunicación industrial estandarizados y ciberseguros (como OPC UA o MQTT) para la visualización de datos en tiempo real, así como el vínculo con sistemas PLM para la gestión del ciclo de vida del producto.
- **Validación de usabilidad y eficiencia:** Realizar estudios experimentales centrados en el factor humano para cuantificar la reducción en la carga cognitiva, el incremento en la eficiencia operativa y la mejora en la curva de aprendizaje, permitiendo así calcular el retorno de inversión para las organizaciones.

Apéndice A

Funciones completas de animaciones en JavaScript

```
// Funcion Traslacion:
//  - nombreModelo: Nombre del modelo CAD al que se le aplicara la traslacion. Es un parametro de entrada.
//  - eje: Eje en el que se realiza la traslacion (por ejemplo, "x", "y", "z"). Es un parametro de entrada.
//  - Avance: Distancia maxima de traslacion a la que se mueve el modelo desde su posicion inicial. Es un
    parametro de entrada.
//  - timingInterval: Intervalo de tiempo en milisegundos entre cada actualizacion de la posicion del modelo. Es
    un parametro de entrada.
//  - sentido: Direccion del movimiento (+ o -) en el eje especificado. Es un parametro de entrada.
//  - cordenadaOriginal: Posicion inicial del modelo en el eje especificado, extraida de ‘parametrosModeloItem‘
    y convertida a un numero.
//  - estado: Variable booleana que controla si el modelo esta en movimiento o si el proceso se debe detener.

$scope.Traslacion = function(nombreModelo, eje, Avance, timingInterval, sentido) {
  // La funcion traslacion retorna una promesa
  return new Promise((resolve, reject) => {
    // Almacena la posicion inicial del modelo en el eje especificado, asegurando que sea un numero
    var cordenadaOriginal = parametrosModeloItem[nombreModelo][eje] - 0;

    // Verifica si el modelo no esta en la posicion inicial; si no, lo devuelve a su posicion original
    if ($scope.view.wdg[nombreModelo][eje] != cordenadaOriginal) {
      $scope.view.wdg[nombreModelo][eje] = cordenadaOriginal;
    }

    // Variable para controlar el estado de movimiento del modelo
    var estado = false;

    // Inicia un intervalo que actualizara la coordenada del modelo en cada ciclo de tiempo definido por
    timingInterval
    var updateCordenada = $interval(function() {

      // Verifica si el boton fue presionado de nuevo y si el modelo esta en su posicion inicial
      // Ademas, revisa si el modelo esta en listaObjetosTranslacion
      if ($scope.view.wdg[nombreModelo][eje] == cordenadaOriginal && listaObjetosTranslacion.includes(
    nombreModelo) == true) {
        // Elimina el nombre del modelo del array listaObjetosTranslacion
        listaObjetosTranslacion.splice(listaObjetosTranslacion.indexOf(nombreModelo), 1);
        // Rechaza la promesa indicando que el boton fue presionado de nuevo
        reject('Se presiono el boton de nuevo');
        // Cancela el intervalo para evitar un ciclo infinito
        $interval.cancel(updateCordenada);
        // Cambia el estado a true para indicar que el movimiento fue cancelado
        estado = true;
      }

      // Si el movimiento no fue cancelado
      if (estado == false) {
        // Verifica si el nombre del modelo no esta en listaObjetosTranslacion
        if (listaObjetosTranslacion.includes(nombreModelo) == false) {
          // Agrega el nombre del modelo al array listaObjetosTranslacion
          listaObjetosTranslacion.push(nombreModelo);
        }

```

```
        // Actualiza la posicion del modelo en el eje especificado, sumando o restando un valor segun el sentido
        $scope.view.wdg[nombreModelo][eje] = $scope.view.wdg[nombreModelo][eje] + Number(sentido + 0.006);

        // Verifica si el modelo ha alcanzado el limite de traslacion definido por Avance
        if ($scope.view.wdg[nombreModelo][eje] > cordenadaOriginal + Avance || $scope.view.wdg[nombreModelo][eje
    ] < cordenadaOriginal - Avance) {
          // Elimina el nombre del modelo del array listaObjetosTranslacion al finalizar el movimiento
          listaObjetosTranslacion.splice(listaObjetosTranslacion.indexOf(nombreModelo), 1);
          // Resuelve la promesa indicando que la traslacion se completo exitosamente
          resolve('Operacion de traslacion exitosa');
          // Cancela el intervalo para evitar un ciclo infinito
          $interval.cancel(updateCordenada);
        }
      }
    }, timingInterval); // Define el intervalo de tiempo para cada iteracion, en milisegundos
  });
};
```

Código A.1 Función completa para la animación de traslación.

```
//---------------Animacion del agua--------------//
$scope.animacionAgua = function(){

  // Oculta panel izquierdo y botones de operacion
  $scope.view.wdg['Sis-M1']['visible'] = false;
  $scope.view.wdg['SisClose']['visible'] = true;                              // Habilita boton de cerrar menu (
    DETENER)
  $scope.view.wdg['Oper_M2']['visible'] = true;
  $scope.view.wdg['ToggleOpen']['visible'] = true;                            // Habilita boton toggle del panel
    Oper_M2
  $scope.view.wdg['ToggleOpen']['value'] = true;                              // Valor inicial de ToggleOpen es
    true

  // Carga el valor de ToggleOpen en ToggleOper para mostrar el menu Oper_M2
  $scope.app.params.ToggleOper = $scope.view.wdg['ToggleOpen']['value'];

  var animacionAgua = function() {
    verificarEstados(); // Verifica los estados actuales de las flechas
    var timingInterval = 10; // Intervalo de tiempo en milisegundos para el movimiento
    var coordenadaOriginalFlechaAgua = parametrosModeloItem['flechaAgua-1']['y']; // Guarda posicion inicial de
    flechaAgua-1
    var distancia = 0; // Almacena distancia recorrida por flechaAgua-1
    var promesa; // Guarda promesa generada por la funcion Traslacion

    var iniciarAnimacion = function() { // Inicia la animacion de agua
      if (estadoAnimacion == true) return false; // Si animacion ya esta en ejecucion, se detiene
      estadoAnimacion = true; // Marca animacion como activa

      var sentido; // Direccion de movimiento de la flecha
      var eje; // Eje en el que se mueve la flecha
      var avance = 0.3; // Avance en cada paso de flechaAgua-1

      promesa = $scope.Traslacion('flechaAgua-1', 'y', avance, timingInterval, '-'); // Llama a funcion de
    traslacion y guarda promesa

      // Ciclo para actualizar posiciones de flechas
      var traslacionFlechas = $interval(function() {
        for (var i = 2; i <= 55; i++) { // Recorre flechas desde flechaAgua-2 hasta flechaAgua-55
          if (detenerAnimacionAgua == true) { // Verifica si animacion debe detenerse
            detenerAnimacionAgua = false;
            $interval.cancel(traslacionFlechas); // Cancela intervalo para detener animacion
            break;
          }

          var flechaAguaN = 'flechaAgua-' + i.toString(); // Convierte i en string para formar el nombre de la
    flecha actual

          if (flechaAguaN in z_Flechas) { // Si flecha debe moverse en eje z
            eje = 'z';
            sentido = z_Flechas[flechaAguaN];
          } else if (parametrosModeloItem[flechaAguaN]['ry'] <= 2) { // Si flecha debe moverse en eje x o y
            var aux = Math.abs(parametrosModeloItem[flechaAguaN]['ry']); // Valor absoluto de ry para decidir
    eje

            if (parametrosModeloItem[flechaAguaN]['ry'] > 0) { // Si ry es positivo, sentido positivo
              sentido = 1;
```

```
            eje = aux == 1 ? 'x' : 'y'; // Si aux es 1, usa eje x, si no, usa eje y
          } else { // Si ry es negativo, sentido negativo
            sentido = -1;
            eje = aux == 1 ? 'x' : 'y'; // Si aux es 1, usa eje x, si no, usa eje y
          }
        }

        // Calcula y actualiza posicion de la flecha actual
        distancia = coordenadaOriginalFlechaAgua - $scope.view.wdg['flechaAgua-1']['y'];
        $scope.view.wdg[flechaAguaN][eje] = parametrosModeloItem[flechaAguaN][eje] + sentido * distancia;
      }

      // Verifica si flechaAgua-1 completo su recorrido
      if ($scope.view.wdg['flechaAgua-1']['y'] <= coordenadaOriginalFlechaAgua - avance) {
        estadoAnimacion = false; // Marca animacion como inactiva
        $interval.cancel(traslacionFlechas); // Cancela ciclo de traslacion para optimizar memoria
        traslacionFlechas = null;
      }
    }, 5); // Ejecuta cada 5 ms
  };

  var animacionAgua = iniciarAnimacion(); // Asigna iniciarAnimacion para evitar fugas de memoria

  // Ciclo para reiniciar animacion cuando la promesa se cumple
  var secuenciaAnimacion = $interval(function() {
    promesa.then((resolve) => {
      animacionAgua = iniciarAnimacion(); // Reinicia animacion
    }).catch((error) => { // Si promesa falla, detiene animacion
      $interval.cancel(secuenciaAnimacion); // Cancela ciclo si se presiona detener
      console.log(error); // Muestra error en consola
    });
  }, 10); // Ejecuta cada 10 ms
}

// Inicia animacion con retardo de 30 ms
$timeout(animacionAgua, 30);
};
```

Código A.2 Función para animar las flechas que representan flujos de agua.

```
// Funcion para verificar estados y actualizar la visualizacion
var verificarEstados = function(bool){
  // Variables importantes que representan el estado de las valvulas y la animacion
  var A = detenerAnimacionAgua;  // Indica si la animacion del agua debe detenerse
  var B = estadoCutValveF;       // Estado de la valvula de corte en el circuito de transferencia (abierta/
    cerrada)
  var C = estadoControlValveF;    // Estado de la valvula de control en el circuito de transferencia (abierta/
    cerrada)
  var D = estadoCutValveR;        // Estado de la valvula de corte en el circuito de retroalimentacion (abierta/
    cerrada)
  var E = estadoControlValveR;    // Estado de la valvula de control en el circuito de retroalimentacion (
    abierta/cerrada)
  var F = TO;                     // Estado general del sistema (posiblemente una senal de encendido)

  var flechaAguaN;                // Variable para iterar sobre las flechas de agua
  for(var j = 1; j <= 55; j++){
    flechaAguaN = 'flechaAgua-' + j.toString(); // Nombre dinamico de la flecha, de flechaAgua-1 a flechaAgua-55

    // Actualizacion de visibilidad de las flechas en funcion del estado de las valvulas y animacion
    if(j >= 1 && j <= 11){
      $scope.view.wdg[flechaAguaN]['visible'] = !A; // Flechas visibles si la animacion no esta detenida
    }
    else if(j >= 12 && j<= 41){
      $scope.view.wdg[flechaAguaN]['visible'] = !A && !(B && C); // Flechas del circuito de transferencia
     visibles si valvulas B y C estan cerradas
    }
    else{
      $scope.view.wdg[flechaAguaN]['visible'] = !A && !(D && E); // Flechas del circuito de retroalimentacion
     visibles si valvulas D y E estan cerradas
    }
  }

  // Control de opacidad de los circuitos en funcion de las valvulas y animacion
  $scope.view.wdg['circuitoTubTran']['opacity'] = 0.3 * !(D && E) + 1 * (!(B && C) || A);
  $scope.view.wdg['circuitoTubRetro']['opacity'] = 0.3 * !(B && C) + 1 * (!(D && E) || A);

```

```
  // Configuracion de botones de circuito en funcion del estado de animacion y valvulas
  $scope.view.wdg['button-3']['visible'] = !(!A && !(B && C));
  $scope.view.wdg['button-4']['visible'] = !(!A && !(D && E));

  // Configuracion de botones de valvulas en funcion del estado del sistema y las valvulas
  $scope.view.wdg['CutVlvFlujo']['visible'] = F && !(!A && !(D && E));
  $scope.view.wdg['CtrlVlvFlujo']['visible'] = F && !(!A && !(D && E));
  $scope.view.wdg['CutVlvRetro']['visible'] = F && !(!A && !(B && C));
  $scope.view.wdg['CtrlVlvRetro']['visible'] = F && !(!A && !(B && C));
  $scope.view.wdg['Switch-Circuits']['visible'] = F;

  // Configuracion de flechas de agua adicionales
  $scope.view.wdg['flechaAgua-18']['visible'] = !A && !B;
  $scope.view.wdg['flechaAgua-19']['visible'] = !A && !B;
  $scope.view.wdg['flechaAgua-20']['visible'] = !A && !B;
  $scope.view.wdg['flechaAgua-21']['visible'] = !A && !C;
  $scope.view.wdg['flechaAgua-22']['visible'] = !A && !C;
  $scope.view.wdg['flechaAgua-23']['visible'] = !A && !C;
  $scope.view.wdg['flechaAgua-46']['visible'] = !A && !D;
  $scope.view.wdg['flechaAgua-47']['visible'] = !A && !D;
  $scope.view.wdg['flechaAgua-48']['visible'] = !A && !D;
  $scope.view.wdg['flechaAgua-49']['visible'] = !A && !E;
  $scope.view.wdg['flechaAgua-50']['visible'] = !A && !E;
  $scope.view.wdg['flechaAgua-51']['visible'] = !A && !E;

  // Indicadores de las valvulas de control
  $scope.view.wdg['3DGauge-1']['visible'] = !A;
  $scope.view.wdg['3DGauge-4']['visible'] = !A;

  // Indicadores de estado de las valvulas en el circuito de transferencia
  $scope.view.wdg['CrossCtrlValveF']['visible'] = !A && C;
  $scope.view.wdg['CrossCutValveF']['visible'] = !A && B;
  $scope.view.wdg['CheckCtrlValveF']['visible'] = !A && !C;
  $scope.view.wdg['CheckCutValveF']['visible'] = !A && !B;
  $scope.view.wdg['CtrlVlvFlujo']['value'] = !A && !C;
  $scope.view.wdg['CutVlvFlujo']['value'] = !A && !B;

  // Indicadores de estado de las valvulas en el circuito de retroalimentacion
  $scope.view.wdg['CheckCtrlValveR']['visible'] = !A && !E;
  $scope.view.wdg['CheckCutValveR']['visible'] = !A && !D;
  $scope.view.wdg['CrossCtrlValveR']['visible'] = !A && E;
  $scope.view.wdg['CrossCutValveR']['visible'] = !A && D;
  $scope.view.wdg['CtrlVlvRetro']['value'] = !A && !E;
  $scope.view.wdg['CutVlvRetro']['value'] = !A && !D;

  $scope.$apply(); // Actualizar vista
};

// Funcion para verificar si ambas valvulas de un circuito estan abiertas y ajustar el circuito visible
var verificarValvulas = function(numero){
  if(numero == 1){
    if(estadoControlValveF && estadoCutValveF){ // Ambas valvulas en el circuito de transferencia estan abiertas
      $scope.mostrarCircuitoRetro(); // Mostrar circuito de retroalimentacion
      $scope.view.wdg['Switch-Circuits']['value'] = false;
      $scope.view.wdg['Switch-Circuits']['label'] = 'Recirculacion';
    }
  } else if (numero == 2){
    if(estadoControlValveR && estadoCutValveR){ // Ambas valvulas en el circuito de retroalimentacion estan
     abiertas
      $scope.mostrarcircuitoTran(); // Mostrar circuito de transferencia
      $scope.view.wdg['Switch-Circuits']['value'] = true;
      $scope.view.wdg['Switch-Circuits']['label'] = 'Transferencia';
    }
  }
};
```

Código A.3 Funciones para verificar y actualizar la visibilidad de las flechas de flujo y los indicadores de las válvulas.

```
//=============Valvulas de TRANSFERENCIA==============================
// Este codigo actualiza los valores ON/OFF de las variables asociadas a las valvulas, lo que permite cambiar
     sus indicadores visuales y cambiar el comportamiento de los flujos de flechas

$scope.btnCutValveF = function(){ // Funcion para cambiar el estado de la valvula de corte F cuando se presiona
     el boton
  estadoCutValveF = !estadoCutValveF; // Se cambia el estado de la valvula de corte F
```

```
  verificarEstados(); // Funcion para verificar y mostrar las actualizaciones en la vista
  verificarValvulas(1); // Funcion para verificar los estados de la valvula de corte y control F
};

$scope.btnCntValveF = function(){ // Funcion para cambiar el estado de la valvula de control F cuando se presiona el boton
  estadoControlValveF = !estadoControlValveF; // Se cambia el estado de la valvula de control F
  verificarEstados(); // Funcion para verificar y mostrar las actualizaciones en la vista
  verificarValvulas(1); // Funcion para verificar los estados de la valvula de corte y control F
};

//=============Valvulas de RECIRCULACION==============================

$scope.btnCntValveR = function(){ // Funcion para cambiar el estado de la valvula de control R cuando se presiona el boton
  estadoControlValveR = !estadoControlValveR; // Se cambia el estado de la valvula de control R
  verificarEstados(); // Funcion para verificar y mostrar las actualizaciones en la vista
  verificarValvulas(2); // Funcion para verificar los estados de la valvula de corte y control R
};

$scope.btnCutValveR = function(){ // Funcion para cambiar el estado de la valvula de corte R cuando se presiona el boton
  estadoCutValveR = !estadoCutValveR; // Se cambia el estado de la valvula de corte R
  verificarEstados(); // Funcion para verificar y mostrar las actualizaciones en la vista
  verificarValvulas(2); // Funcion para verificar los estados de la valvula de corte y control R
};
```

Código A.4 Funciones para cambiar los estados de las válvulas.

```
$scope.destacarInstrumento = function(nombreInstrumento){
  // Inicia la funcion para resaltar un instrumento especifico basado en su nombre

  if(mostrarPopup){ // Verifica si debe mostrarse el popup de detalles
    $scope.default(); // Restaura opacidades predeterminadas para todos los instrumentos

    // Recorre la lista de instrumentos para ajustar opacidades segun el nombre proporcionado
    for(var i = 0; i < listaInstrumentos.length; i++){
      // Compara cada instrumento con el nombre proporcionado
      if(listaInstrumentos[i] == nombreInstrumento){
        // Si el instrumento es 'RotametroF', ajusta opacidad completa para su placa de nivel
        if(listaInstrumentos[i] == 'RotametroF'){
          $scope.view.wdg['placaNivelF']['opacity'] = 1; // Muestra 'placaNivelF' al 100%
        } else if(listaInstrumentos[i] == 'RotametroR'){
          // Si el instrumento es 'RotametroR', ajusta opacidad completa para su placa de nivel
          $scope.view.wdg['placaNivelR']['opacity'] = 1; // Muestra 'placaNivelR' al 100%
        }
      } else {
        // Si el instrumento no coincide con el nombre dado, reduce su opacidad
        $scope.view.wdg[listaInstrumentos[i]]['opacity'] = 0.5; // Ajusta opacidad al 50% para instrumentos no destacados
      }
    }

    // Configura variables para el efecto de parpadeo del instrumento seleccionado
    var opacity = 0; // Inicia opacidad auxiliar en 0 para crear el efecto de parpadeo
    var sentido = '+'; // Direccion inicial de cambio de opacidad (aumentando)
    var instrumento = nombreInstrumento + 'Aux'; // Define nombre auxiliar del instrumento seleccionado

    $scope.view.wdg[instrumento]['visible'] = true; // Muestra el instrumento auxiliar para el parpadeo
    parpadeoInst = true; // Activa el indicador de parpadeo

    // Inicia un intervalo que controla el parpadeo del instrumento auxiliar
    parpadeoInstrumento = $interval(function(){
      if(mostrarPopup){ // Comprueba si el popup sigue activo
        opacity = $scope.view.wdg[instrumento]['opacity']; // Obtiene la opacidad actual del instrumento auxiliar

        // Controla la direccion del parpadeo (aumentar o disminuir opacidad)
        if(sentido == '+'){
          if(opacity < 1){
            $scope.view.wdg[instrumento]['opacity'] = opacity + 0.01; // Incrementa opacidad gradualmente
          } else {
            sentido = '-'; // Cambia a disminuir cuando la opacidad alcanza 1
          }
        } else {
          if(opacity > 0){
```

```
            $scope.view.wdg[instrumento]['opacity'] = opacity - 0.01; // Disminuye opacidad gradualmente
          } else {
            sentido = '+'; // Cambia a aumentar cuando la opacidad llega a 0
          }
        }
      } else {
        // Si el popup se cierra, cancela el efecto de parpadeo
        $interval.cancel(parpadeoInstrumento);
      }
    }, 10); // Configura la frecuencia del parpadeo cada 10 ms
  }
};

// Funcion para restaurar los valores de opacidad predeterminados de todos los instrumentos
$scope.default = function(){
  $interval.cancel(parpadeoInstrumento); // Detiene cualquier parpadeo en curso

  var instrumento; // Variable temporal para nombres de instrumentos auxiliares

  // Recorre todos los instrumentos para restaurar sus opacidades y visibilidades
  for(var i = 0; i < listaInstrumentos.length; i++){
    $scope.view.wdg[listaInstrumentos[i]]['opacity'] = 1; // Restaura opacidad al 100% para cada instrumento

    if(i >= 5){ // Si el indice supera 5, trabaja con el instrumento auxiliar
      instrumento = listaInstrumentos[i] + 'Aux'; // Define nombre del instrumento auxiliar
      $scope.view.wdg[instrumento]['opacity'] = 0; // Oculta la opacidad del auxiliar
      $scope.view.wdg[instrumento]['visible'] = false; // Oculta visualmente el instrumento auxiliar
    }
  }
};
```

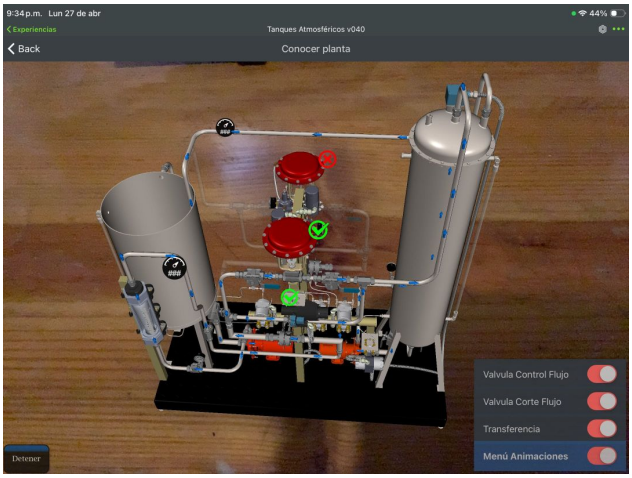
Código A.5 Animación para destacar una parte de un modelo.

Apéndice B

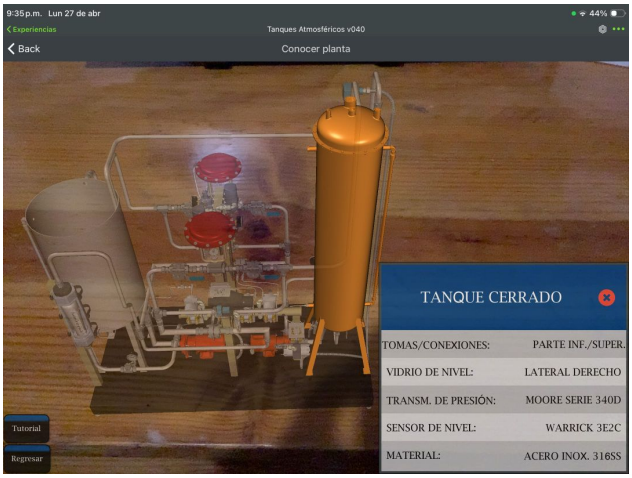
Fotografías de la puesta en operación de la experiencia desarrollada

Figura B.1 Vista inicial (portada) de la experiencia. [1]

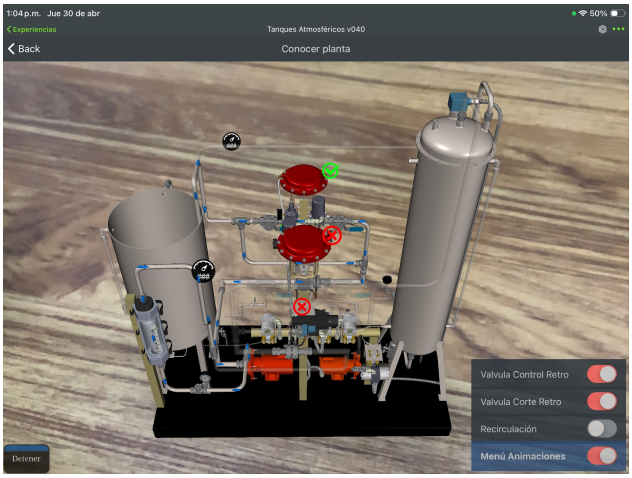
[1] Captura de pantalla de autoría propia.
[2] Capturas de pantalla de autoría propia.
[3] Capturas de pantalla de autoría propia.



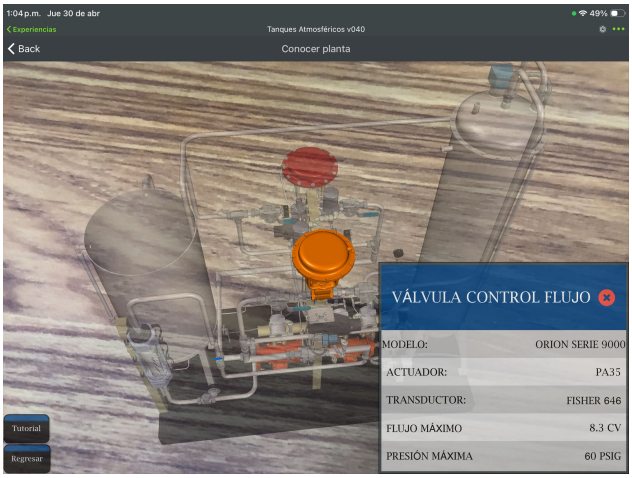
(a) Circuito de transferencia en AR.



(b) Tanque destacado en la opción “Capacitación”.

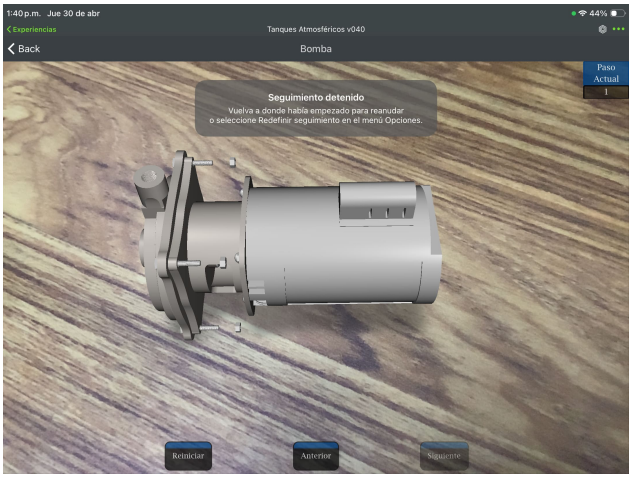


(c) Circuito de recirculación en AR.

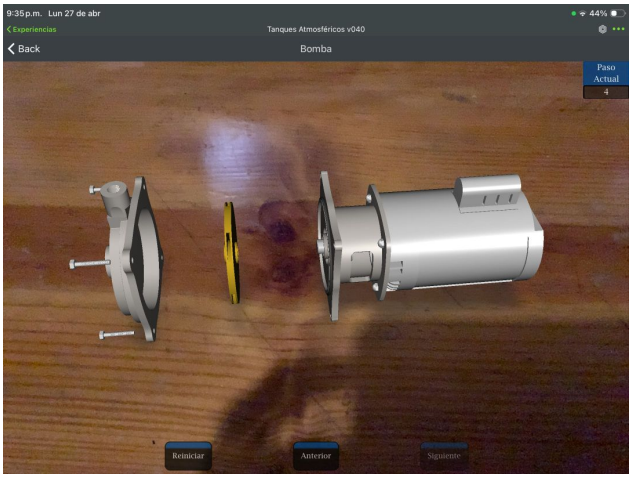


(d) Válvula de control en la opción “Capacitación”.

Figura B.2 *Digital overlays* de las funcionalidades en la vista “Conocer Planta” [2]

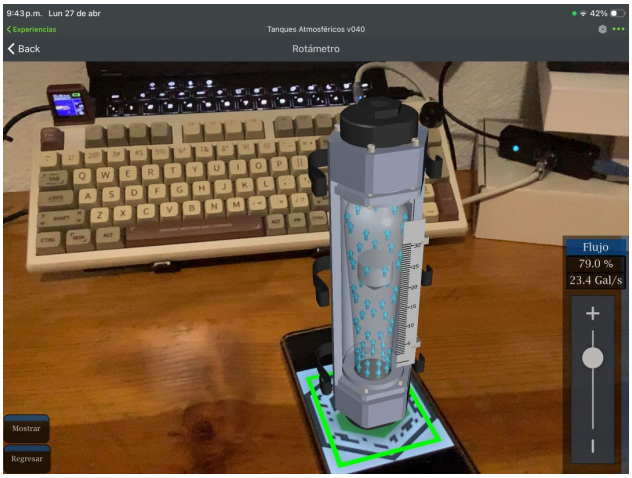


(a) Estado inicial de la secuencia de una bomba en AR.

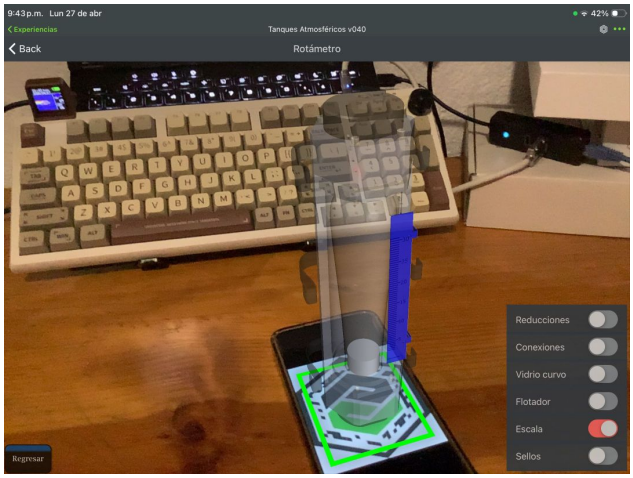


(b) Paso 4 de la secuencia sobre una bomba en AR.

Figura B.3 Despiece de una bomba centrifuga del sistema de tanques en AR. [3]

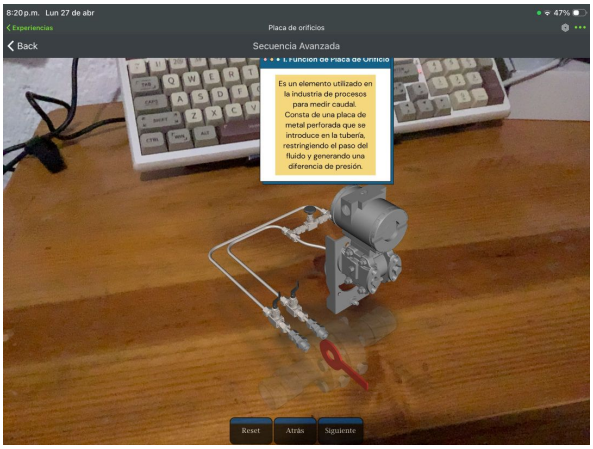


(a) *Digital overlay* en la opción "Funcionamiento".

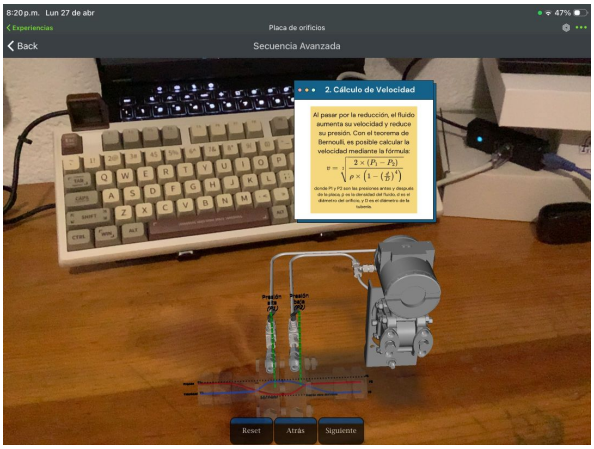


(b) *Digital overlay* en la opción "Ver componentes".

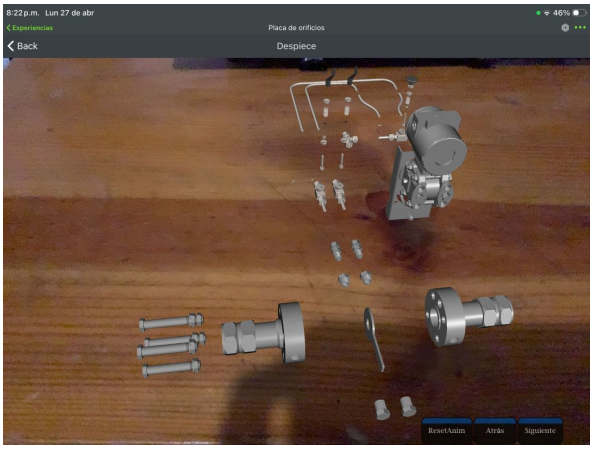
Figura B.4 Vista en AR del rotámetro del sistema de tanques. [4]



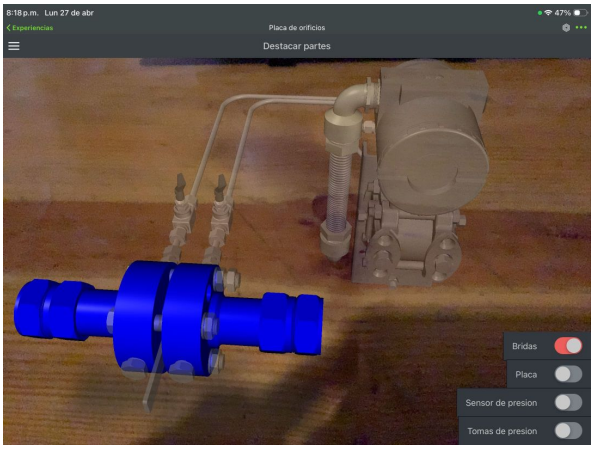
(a) Paso 1 de la secuencia de la placa de orificios.



(b) Paso 2 de la secuencia de la placa de orificios.



(c) Paso 3 de la secuencia de la placa de orificios.



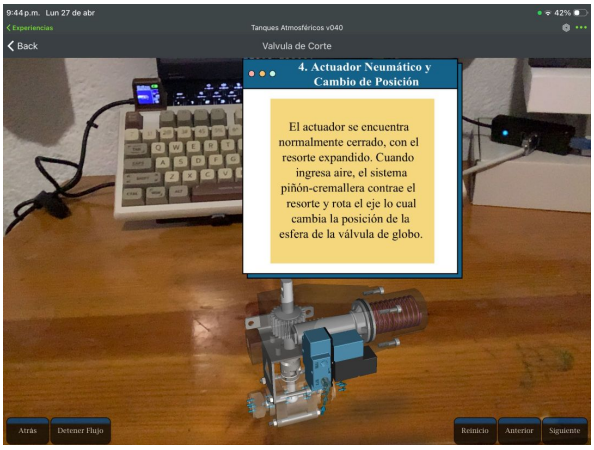
(d) *Digital overlay* de la opción `Despiece`.

Figura B.5 *Digital overlays* de las funcionalidades en la vista `Conocer Planta`. [5]

[4]Capturas de pantalla de autoría propia.

(a) Paso 1 de la secuencia en AR.

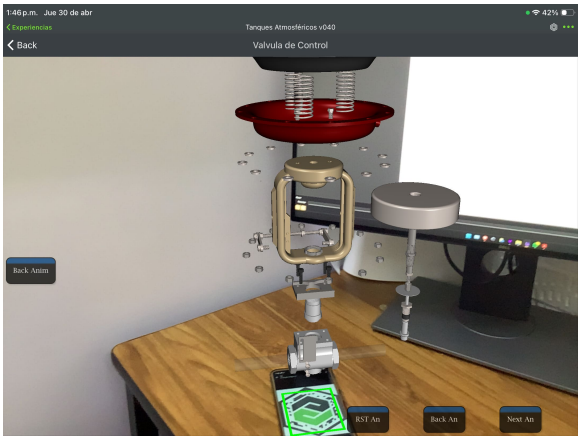
(b) Paso 3 de la secuencia en AR.



(c) Paso 4 de la secuencia en AR.

(d) Despiece de una válvula en AR.

Figura B.6 *Digital overlays* de las funcionalidades en la vista `Válvula de corte`. [6]



(a) Despiece de una válvula en AR.

(b) Paso 1 de la secuencia sobre una válvula en AR

Figura B.7 *Digital overlays* de las funcionalidades en la vista `Válvula de control`. [7]

[5] Capturas de pantalla de autoría propia.
[6] Capturas de pantalla de autoría propia.
[7] Capturas de pantalla de autoría propia.

Referencias

[Alvarez, 2020] Alvarez, M. A. (2020). Introducción a css. Consultado el 4 de septiembre de 2024 https://desarrolloweb.com/articulos/181.php. (Citado en página 40.)

[AngularJS, 2022] AngularJS (2022). Developer guide. https://docs.angularjs.org/guide. Accedido: 1 de septiembre de 2024. (Citado en páginas 38, 39 y 40.)

[Azuma, 1997] Azuma, R. T. (1997). A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, *6*(4), 355–385, https://doi.org/10.1162/pres.1997.6.4.355 https://doi.org/10.1162/pres.1997.6.4.355. (Citado en páginas 17 y 18.)

[BBCMundo, 2011] BBCMundo (2011). Realidad aumentada: ¿revolución o promesa vana? Accedido el 26 de marzo de 2024 https://www.bbc.com/mundo/noticias/2011/05/110505_1653_realidad_aumentada_utilidad_negocios_dc. (Citado en página 6.)

[BMW Group, 2020] BMW Group (2020). Munich pilot plant: Bmw group uses augmented reality in prototyping. Publicado en PressClub Global, 18.09.2020, Consultado en línea https://www.press.bmwgroup.com/global/article/detail/T0317125EN/munich-pilot-plant:-bmw-group-uses-augmented-reality-in-prototyping. (Citado en páginas 2 y 10.)

[Chen, *et al.*, 2019] Chen, Y., Wang, Q., Chen, H., Song, X., Tang, H., & Tian, M. (2019). An overview of augmented reality technology. *1237*(2), 022082, https://doi.org/10.1088/1742-6596/1237/2/022082 https://dx.doi.org/10.1088/1742-6596/1237/2/022082. (Citado en página 21.)

[Comisión Electrotécnica Internacional, 2010] Comisión Electrotécnica Internacional (2010). *IEC 61508 Versión Comentada, Edición 2*. Norma IEC 61508, Comisión Electrotécnica Internacional, Ginebra, CH. Accedido el 17 de mayo de 2024 https://share.ansi.org/Shared%20Documents/News%20and%20Publications/Other%20Documents/IEC%2061508%20Commented%20Version.pdf. (Citado en páginas 3 y 13.)

[Coronado, *et al.*, 2023] Coronado, E., Itadera, S., & Ramirez-Alpizar, I. G. (2023). Integrating virtual, mixed, and augmented reality to human–robot interaction applications using game engines: A brief review of accessible software tools and frameworks. *Applied Sciences*, *13*(3), https://doi.org/10.3390/app13031292 https://www.mdpi.com/2076-3417/13/3/1292. (Citado en página 11.)

[Corrales, 2019] Corrales, J. A. (2019). Interfaz de usuario o ui: ¿qué es y cuáles son sus características? Accessed: 2024-08-15 https://rockcontent.com/es/blog/interfaz-de-usuario/. (Citado en páginas 20 y 21.)

[Craig, 2013] Craig, A. (2013). *Understanding Augmented Reality: Concepts and Applications*. Elsevier Science https://books.google.es/books?id=7_O5LaIC0SwC. (Citado en páginas 1, 17, 18 y 19.)

[Dan Miller, 2024] Dan Miller, Tim Mowrer, B. W. (2024). Unity's handheld ar ecosystem: Ar foundation, arcore and arkit. Accessed: 2024-06-14 https://blog.unity.com/technology/unitys-handheld-ar-ecosystem-ar-foundation-arcore-and-arkit. (Citado en página 6.)

[De Pace, *et al.*, 2018] De Pace, F., Manuri, F., & Sanna, A. (2018). Augmented reality in industry 4.0. *Am. J. Comput. Sci. Inf. Technol*, *6*(1), 17. (Citado en páginas 10, 13 y 26.)

[Duda*, et al.*, 2022] Duda, J., Oleszek, S., & Santarek, K. (2022). *Product Lifecycle Management (PLM) in the Context of Industry 4.0*, (pp. 171–185). (Citado en página 21.)

[Erazo-Arteaga, 2022] Erazo-Arteaga, V. A. (2022). El diseño, la manufactura y análisis asistido por computadora (CAD/CAM/CAE) y otras técnicas de fabricación digital en el desarrollo de productos en América Latina. *Información tecnológica*, *33*, 297–308, https://doi.org/10.4067/S0718-07642022000200297 http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642022000200297&nrm=iso. (Citado en página 27.)

[Escaño González*, et al.*, 2019] Escaño González, J., NUEVO GARCIA, A., & GARCÍA CABALLERO, J. (2019). *Integración de sistemas de automatización industrial*. Ediciones Paraninfo, S.A https://books.google.com.mx/books?id=gj2dDwAAQBAJ. (Citado en páginas 3 y 13.)

[Fabrício Herpich, 2017] Fabrício Herpich, Renan Luigi Martins Guarese, L. M. R. T. (2017). A comparative analysis of augmented reality frameworks aimed at the development of educational applications. *Creative Education*, *8*(9) www.scirp.org/journal/ce. (Citado en página 11.)

[Formlabs, 2023] Formlabs (2023). Choosing the best 3d cad software: A comprehensive guide. https://formlabs.com/blog/cad-software/. Accedido: 23 de agosto de 2024. (Citado en página 29.)

[Garza*, et al.*, 2013] Garza, L. E., *, et al.* (2013). Augmented reality application for the maintenance of a flapper valve of a fuller-kynion type m pump. *Procedia Computer Science*, *25*, 154–160, https://doi.org/https://doi.org/10.1016/j.procs.2013.11.019. 2013 International Conference on Virtual and Augmented Reality in Education https://www.sciencedirect.com/science/article/pii/S1877050913012246. (Citado en páginas 4 y 13.)

[Gavilanes, 2020] Gavilanes, D. (2020). Review de apptim: Pruebas de performance en apps nativas. Online. web de Federico Toledo https://federico-toledo.com/review-de-apptim-pruebas-de-performance-en-apps-nativas/. (Citado en página 81.)

[GodoyJr, 2020] GodoyJr, C. (2020). Augmented reality and gamification: A framework for developing supplementary learning tool. *International Journal of Computing Sciences Research*, *5*. (Citado en página 11.)

[Google, 2024] Google (2024). Arcore supported devices. Consulted on August 15, 2024 https://developers.google.com/ar/devices. (Citado en página 22.)

[Heaney, 2024] Heaney, D. (2024). Apple vision pro specifications and features. Accessed: 2024-08-18 https://www.uploadvr.com/apple-vision-pro-specs/. (Citado en página 24.)

[Hector, 2023] Hector, H. (2023). Meta quest 3 review - the new best vr headset for most people. *TechRadar* https://www.techradar.com/computing/virtual-reality-augmented-reality/hands-on-meta-quest-3-review. (Citado en página 24.)

[Herdina, 2020] Herdina, M. (2020). Augmented reality disappeared from gartner's hype cycle – what's next? *ARPost*. Disponible en ARPost. [Consultado el 26 de marzo de 2024] https://arpost.co/2020/09/25/augmented-reality-gartners-hype-cycle/. (Citado en páginas 6 y 13.)

[Hou*, et al.*, 2013] Hou, L., Wang, X., & Bernold, L. (2013). Using animated augmented reality to cognitively guide assembly. *Journal of Computing in Civil Engineering*, https://doi.org/10.1061/(ASCE)CP.1943-5487.0000184. (Citado en páginas 4 y 13.)

[IBM, 2024] IBM (2024). ¿qué es la industria 4.0? Accedido el 20 de agosto de 2024 https://www.ibm.com/mx-es/topics/industry-4-0. (Citado en página 25.)

[Integral Innovation Experts, 2024] Integral Innovation Experts (2024). Crea y publica experiencias industriales de realidad aumentada a escala empresarial con vuforia studio. Accessed: 2024-06-13 https://integralplm.com/vuforia-studio. (Citado en página 30.)

[Kamble, *et al.*, 2018] Kamble, S. S., Gunasekaran, A., & Gawankar, S. A. (2018). Sustainable industry 4.0 framework: A systematic literature review identifying the current trends and future perspectives. *Process Safety and Environmental Protection*, *117*, 408–425, https://doi.org/https://doi.org/10.1016/j.psep.2018.05.009 https://www.sciencedirect.com/science/article/pii/S0957582018301629. (Citado en páginas 13, 21, 25 y 26.)

[Kljun, *et al.*, 2020] Kljun, M., Geroimenko, V., & Čopič Pucihar, K. (2020). *Augmented Reality in Education: Current Status and Advancement of the Field*, (pp. 3–21). Springer International Publishing: Cham https://doi.org/10.1007/978-3-030-42156-4_1. (Citado en página 11.)

[Larson, 2022] Larson, M. (2022). Interactive lecture: Small changes for improved engagement and learning. (Citado en páginas 4 y 13.)

[Lenovo, 2020] Lenovo (2020). Thinkreality a3 smart glasses. https://www.lenovo.com/es/es/p/smart-devices/virtual-and-augmented-reality/thinkreality/thinkreality-a3-pc-edition/wmd00000500#tech_specs. Accedido: 18 de agosto de 2024. (Citado en página 24.)

[Longo, *et al.*, 2017] Longo, F., Nicoletti, L., & Padovano, A. (2017). Smart operators in industry 4.0: A human-centered approach to enhance operators' capabilities and competencies within the new smart factory context. *Computers & Industrial Engineering*, *113*, 144–159, https://doi.org/10.1016/j.cie.2017.09.016 https://www.sciencedirect.com/science/article/pii/S0360835217304291. (Citado en páginas 9 y 13.)

[Magic Leap, 2024] Magic Leap (2024). Magic leap 2. https://www.magicleap.com/magic-leap-2. Consultado: 18 de agosto de 2024. (Citado en página 24.)

[Marino, *et al.*, 2021] Marino, E., Barbieri, L., Colacino, B., Fleri, A. K., & Bruno, F. (2021). An augmented reality inspection tool to support workers in industry 4.0 environments. *Computers in Industry*, *127*, 103412, https://doi.org/10.1016/j.compind.2021.103412 https://www.sciencedirect.com/science/article/pii/S0166361521000191. (Citado en página 10.)

[Mayer y Fiorella, 2014] Mayer, R. E. & Fiorella, L. (2014). *Principles for Reducing Extraneous Processing in Multimedia Learning: Coherence, Signaling, Redundancy, Spatial Contiguity, and Temporal Contiguity Principles*, (pp. 279–315). Cambridge Handbooks in Psychology. Cambridge University Press. (Citado en páginas 9 y 13.)

[Mendoza-Ramírez, *et al.*, 2023] Mendoza-Ramírez, C. E., Tudon-Martinez, J. C., Félix-Herrán, L. C., Lozoya-Santos, J. d. J., & Vargas-Martínez, A. (2023). Augmented reality: Survey. *Applied Sciences*, *13*(18), https://doi.org/10.3390/app131810491 https://www.mdpi.com/2076-3417/13/18/10491. (Citado en páginas 19 y 22.)

[Microsoft, 2024] Microsoft (2024). Hololens 2 - learn about hololens 2 features and review technical specs. https://www.microsoft.com/en-us/hololens. Accessed: 2024-08-18. (Citado en página 24.)

[Mobley, 2001] Mobley, R. (2001). Plant engineer's handbook https://books.google.com.pa/books?id=9YkqH5HgSgwC. (Citado en páginas 2 y 45.)

[Morales Méndez y del Cerro Velázquez, 2024] Morales Méndez, G. & del Cerro Velázquez, F. (2024). Augmented reality in industry 4.0 assistance and training areas: A systematic literature review and bibliometric analysis. *Electronics*, *13*(6). (Citado en páginas 9, 13 y 25.)

[Moreno, 2001] Moreno, E. G. (2001). *Automatizaciónn de procesos industriales*. Alfaomega Valencia. (Citado en páginas 3 y 13.)

[Mourtzis, *et al.*, 2017] Mourtzis, D., Vlachou, K., Zogopoulos, V., & Xanthi, F. (2017). : (pp. 354–362). (Citado en páginas 9, 21, 22 y 24.)

[Mozilla Developer Network (MDN), 2024a] Mozilla Developer Network (MDN) (2024a). Css basics - learn web development. https://developer.mozilla.org/es/docs/Learn/CSS. Último acceso: 4 de septiembre de 2024. (Citado en páginas 41, 43 y 44.)

[Mozilla Developer Network (MDN), 2024b] Mozilla Developer Network (MDN) (2024b). Javascript — dynamic client-side scripting. https://developer.mozilla.org/en-US/docs/Learn/JavaScript. Último acceso: 4 de septiembre de 2024. (Citado en páginas 35, 36 y 37.)

[Nxrev, 2024] Nxrev (2024). Cad platform comparison: Creo vs fusion 360. Accedido: 26 de agosto de 2024 https://nxrev.com/2022/01/creo-vs-fusion-360/. (Citado en página 29.)

[Pachhandara, 2022] Pachhandara, N. (2022). Performance measurement approaches applied to ar/vr devices. Online; accessed 21-September-2024. OptoFidelity Blog https://www.optofidelity.com/insights/blogs/performance-measurement-approaches-applied-to-ar-vr-devices. (Citado en página 81.)

[Papakostas, *et al.*, 2022] Papakostas, C., Troussas, C., Krouska, A., & et al. (2022). User acceptance of augmented reality welding simulator in engineering training. *Educ Inf Technol*, *27*, 791–817, https://doi.org/10.1007/s10639-020-10418-7 https://doi.org/10.1007/s10639-020-10418-7. (Citado en páginas 9 y 13.)

[Parada, 2021] Parada, M. (2021). Qué es angular. https://openwebinars.net/blog/que-es-angular-2021/. Accedido: 28 de agosto de 2024. (Citado en página 38.)

[Peterson, 2021] Peterson, K. (2021). Learning augmented reality on ios devices using swiftui, arkit, and realitykit. Published in Better Programming. Accedido el 13-06-2024 https://betterprogramming.pub/learning-augmented-reality-on-ios-devices-using-swiftui-arkit-and-realitykit-d12321d1addd. (Citado en página 6.)

[Pitser, 2021] Pitser, B. (2021). Ar is easy: Vuforia studio best practices to creating ar experiences. Publicado en el sitio web de PTC, consultado el 6 de septiembre de 2024 https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://community.ptc.com/sejnu66972/attachments/sejnu66972/studio/6099/2/Vuforia%2520Studio%2520-%2520Best%2520Practices_SEP2018_PDF.PDF&ved=2ahUKEwjRtoOUs9KIAxVgIkQIHROkDgAQFnoECBYQAQ&usg=AOvVaw1vfSM8Kmz8AR_N4wOAFaDb. (Citado en páginas 49, 77 y 78.)

[PTC, 2019] PTC (2019). Vuforia studio product brief. Accessed: 2024-06-13 https://www.ptc.com/-/media/Files/PDFs/Augmented-Reality/Vuforia-Studio-Product-Brief-Feb-2019.pdf. (Citado en página 13.)

[PTC, 2021] PTC (2021). Howden case study: Employs scalable mixed reality solutions to enhance their customers' overall experience. Publicado en el sitio web de PTC, consultado el 4 de septiembre de 2024 https://ptc-p-001.sitecorecontenthub.cloud/api/public/content/eef3ea523e2c440e86127260b32adbfa. (Citado en página 11.)

[PTC, 2024a] PTC (2024a). Bienvenido al centro de ayuda de vuforia studio. https://support.ptc.com/help/vuforia/studio/es/. Accedido: 28 de agosto de 2024. (Citado en páginas 33, 34, 35, 38, 49 y 71.)

[PTC, 2024b] PTC (2024b). Creo: Solución cad 3d para innovación en diseño de productos. Accedido: 26 de agosto de 2024 https://www.ptc.com/es/products/cad/creo. (Citado en página 29.)

[PTC, 2024c] PTC (2024c). Desarrollo de contenido de realidad aumentada con vuforia engine. https://www.ptc.com/es/products/vuforia/vuforia-engine. Accedido el 13-06-2024. (Citado en páginas 6 y 30.)

[PTC, 2024d] PTC (2024d). Onshape vision app. https://www.onshape.com/en/features/onshape-vision/. Accedido: 23 de agosto de 2024. (Citado en página 29.)

[Ramirez, 1992] Ramirez, G. O. J. (1992). *CAD diseño asistido por computadora*. Tesis de licenciatura, Universidad Nacional Autónoma de México, Escuela Nacional de Estudios Profesionales Aragón, México. Sustentante: Jimenez Ramirez, Gerardo Octavio. Asesor: González Maxinez, David Jaime. Clasificación: 001-41132-J1-1992-2 https://hdl.handle.net/20.500.14330/TES01000187566. (Citado en página 27.)

[Raza, 2019] Raza, M. (2019). What's ar cloud? the augmented reality cloud explained. *BMC Blogs*. Accessed: 2024-08-15 https://www.bmc.com/blogs/augmented-reality-cloud/. (Citado en páginas 21 y 22.)

[Redondo, 2012] Redondo, D. A. (2012). Realidad aumentada. *España: Universidad Carlos III de Madrid*. (Citado en página 19.)

[Rockwell Automation, 2023] Rockwell Automation (2023). Innovación, productividad y sostenibilidad para la fabricación del futuro. Presentación en PDF. Consultado el 13 de agosto de 2024. (Citado en páginas 5, 13, 25 y 26.)

[Saldamando, 2024] Saldamando, M. (2024). How augmented reality is changing quality control for carmakers. Publicado en México Business News, Tue, 05/28/2024 - 08:00, Consultado en línea https://mexicobusiness.news/automotive/news/how-augmented-reality-changing-quality-control-carmakers. (Citado en página 10.)

[Sanchis Llopis, *et al.*, 2010] Sanchis Llopis, R., Romero Pérez, J. A., Vicent Ariño, C., & others (2010). *Automatización industrial*. Universitat Jaume I. (Citado en páginas 2 y 13.)

[Santamaría, 2024] Santamaría, M. G. (2024). *Implementación del gemelo digital de un proceso de tanques atmosféricos interconecados*. Tesis de licenciatura, Facultad de Ingeniería, Universidad Nacional Autónoma de México (UNAM), Ciudad Universitaria, Cd.Mx., México. Sustentante: Miguel García Santamaría. Asesor: Hoover Mujica Ortega. (Citado en páginas 8 y 46.)

[Shrivastava, 2021] Shrivastava, A. (2021). Augmented reality. https://www.linkedin.com/pulse/augmented-reality-ayushi-shrivastava. Accedido el 8 de febrero de 2024. (Citado en páginas 5 y 6.)

[SourceForge, 2024] SourceForge (2024). Catia vs. freecad vs. autodesk fusion 360 vs. inventor comparison chart. Accedido: 26 de agosto de 2024 https://sourceforge.net/software/compare/CATIA-vs-FreeCAD-vs-Fusion-360-vs-Inventor/. (Citado en página 29.)

[Suárez, *et al.*, 2019] Suárez, J. C., Salazar, F. F., Nava, I. F., & Hernández, R. H. (2019). Industry 4.0 and digital manufacturing: A design method applying reverse engineering. *Ingeniería*, *24*, 6–28, https://doi.org/10.14483/23448393.13821 http://www.scielo.org.co/scielo.php?script=sci_arttext&pid=S0121-750X2019000100006&nrm=iso. (Citado en página 27.)

[Telefónica, 2011] Telefónica, F. (2011). *Realidad Aumentada: una nueva lente para ver el mundo*. Fundación Telefónica https://books.google.es/books?id=OXHmCgAAQBAJ. (Citado en páginas 1, 2 y 17.)

[Thees, *et al.*, 2020] Thees, M., Kapp, S., Strzys, M., Beil, F., Lukowicz, P., & Kuhn, J. (2020). Effects of augmented reality on learning and cognitive load in university physics laboratory courses. *Computers in Human Behavior*, *108*, 106316, https://doi.org/10.1016/j.chb.2020.106316. (Citado en páginas 9 y 13.)

[Tinoco y Solís, 2014] Tinoco, E. & Solís, I. (2014). *Programación Web con CSS, JavaScript, PHP y AJAX*. Iván Soria Solís https://books.google.es/books?id=QRG-CQAAQBAJ. (Citado en páginas 40, 41, 42 y 43.)

[Tortora, *et al.*, 2021] Tortora, A. M. R., Di Pasquale, V., Franciosi, C., Miranda, S., & Iannone, R. (2021). The role of maintenance operator in industrial manufacturing systems: Research topics and trends. *Applied Sciences*, *11*(7). (Citado en páginas 7, 8, 9 y 13.)

[Towne, 1985] Towne, D. M. (1985). *Cognitive Workload in Fault Diagnosis*. Technical Report TR-107, University of Southern California, Behavioral Technology Labs. (Citado en página 4.)

[Uva*, et al.*, 2018] Uva, A. E., Gattullo, M., Manghisi, V. M., Spagnulo, D., Cascella, G. L., & Fiorentino, M. (2018). Evaluating the effectiveness of spatial augmented reality in smart manufacturing: a solution for manual working stations. *The International Journal of Advanced Manufacturing Technology*, *94*(1), 509–521, https://doi.org/10.1007/s00170-017-0846-4 https://doi.org/10.1007/s00170-017-0846-4. (Citado en página 9.)

[Van Krevelen y Poelman, 2010] Van Krevelen, R. & Poelman, R. (2010). A survey of augmented reality technologies, applications and limitations. *International Journal of Virtual Reality (ISSN 1081-1451)*, *9*, 1, https://doi.org/10.20870/IJVR.2010.9.2.2767. (Citado en página 21.)

[Veinott y Kanki, 1995] Veinott, E. S. & Kanki, B. G. (1995). Identifying human factors issues in aircraft maintenance operations. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 39 (pp. 950–950).: SAGE Publications Sage CA: Los Angeles, CA. (Citado en páginas 4 y 13.)

[Vuforia Engine Team, 2024] Vuforia Engine Team (2024). Vuforia engine overview. Accessed: 2024-08-02 https://developer.vuforia.com/library/getting-started/vuforia-features. (Citado en páginas 12, 13 y 30.)

[Yngchen, 2014] Yngchen (2014). Project 4: Augmented reality instruction (instructar). Of Mechanics and Code (WordPress). Consultado en agosto de 2024 https://ofmechanicsandcode.wordpress.com/2014/12/26/project-4-augmented-reality-instruction-instructar/. (Citado en página 20.)