



UNIVERSIDADE DA CORUÑA

Facultade de Informática

Máster Universitario en Enxeñaría en Informática

---

TRABALLO FIN DE MÁSTER

# VirManDC: Herramienta de monitorización de centros de procesamiento de datos mediante tecnología holográfica

---

**Estudiante:** Adrián Xuíz García  
**Dirección:** Ángel Gómez García  
Alejandro Manuel Mosteiro Vázquez

A Coruña, Septiembre de 2023.



*Dedicado a mi familia, a mi pareja y en especial a mis abuelos, gracias por todo*



## **Agradecimientos**

---

A mis directores Ángel y Alejandro, por ser siempre una fuente de inspiración, conocimiento y críticas constructivas.

Al CITIC, por la disponibilidad de recursos y la oportunidad laboral que condujo a la elaboración de este TFM.

A mi familia, por el incansable apoyo y conseguir que llegara hasta aquí.

A Sara, por apoyarme y darme ánimos cuando más lo necesito.



## **Resumen**

La monitorización de los Centros de Datos es una de las principales tareas de los administradores de sistemas, siendo necesaria una supervisión constante para tratar de detectar y corregir lo antes posible cualquier incidencia que se pueda producir. Esta gestión de los activos de un CPD suele conllevar una alta dedicación horaria y en la mayoría de casos requiere de atención in situ, además de un buen conocimiento de la estructura y organización del CPD. Sin embargo, las características ambientales de un CPD, con unas condiciones de temperatura y humedad bajas para el funcionamiento óptimo de los equipos, y con unos niveles de ruido elevados, hacen que sea recomendable reducir lo máximo posible las estancias del personal técnico en su interior.

Las tecnologías holográficas más modernas permiten mostrar e incluso interactuar con los elementos virtuales, dotando al entorno virtual de realismo e inmersión. Estas características hacen de las plataformas de realidad aumentada y virtual una herramienta novedosa con un gran potencial que ya se utiliza actualmente en ámbitos médicos, ingeniería naval, mecánica e industrial.

Como objetivo principal de este trabajo se busca la implementación de una aplicación de monitorización de un CPD utilizando como dispositivo holográfico las gafas de realidad mixta Microsoft HoloLens 2. Para ello se representará holográficamente el CPD tal y como es en la realidad, permitiendo imitar la localización real del sitio de trabajo y facilitando la comprensión del técnico en cuanto a la posición de los servidores supervisados. Además, se podrá interactuar con el modelo virtual y acceder a cada servidor físico por separado, permitiendo mostrar al usuario una gran cantidad de información útil acerca del estado de ese servidor en tiempo real, e incluso mostrando los datos correspondientes a cada servidor virtual asignando a una máquina específica. Por último, será posible realizar diferentes acciones de utilidad técnica sobre los propios servidores, como conectarse por protocolo seguro de comunicación para la ejecución de comandos o enviar una señal de reinicio a la máquina.

## **Abstract**

Data Center monitoring is one of the main tasks of system administrators, being necessary a constant supervision to try to detect and fix as soon as possible any incident that may occur. This management of the assets of a data center usually involves a high time commitment and in most cases requires on-site attention, as well as a good knowledge of the structure and organization of the data center. However, the environmental characteristics of a data center, with low temperature and humidity conditions for optimal equipment operation, and high

---

noise levels, make it advisable to reduce as much as possible the time technical personnel spend inside the data center.

The latest holographic technologies make it possible to display and even interact with the virtual elements, giving the virtual environment realism and immersion. These characteristics make augmented and virtual reality platforms a novel tool with great potential that is already being used in medical, naval engineering, mechanical and industrial fields.

The main objective of this work is the implementation of a data center monitoring application using Microsoft HoloLens 2 mixed reality glasses as a holographic device. For this purpose, the data center will be represented holographically as it is in reality, allowing to mimic the real location of the work site and facilitating the understanding of the technician regarding the position of the monitored servers. In addition, it will be possible to interact with the virtual model and access each physical server separately, allowing to show the user a great amount of useful information about the status of that server in real time, and even showing the data corresponding to each virtual server assigned to a specific machine. Finally, it will be possible to perform various technically useful actions on the servers themselves, such as connecting via secure communication protocol for command execution or sending a reboot signal to the machine.

**Palabras clave:**

- VirManDC
- CPD
- Centro Procesamiento Datos
- Monitorización
- Zabbix
- Unity
- MRTK
- HoloLens 2
- Servidores
- Realidad Virtual
- Realidad Aumentada
- Realidad Mixta
- Modelos 3D

**Keywords:**

- VirManDC
- DC
- Data Center
- Monitoring
- Zabbix
- Unity
- MRTK
- HoloLens 2
- Servers
- Virtual Reality
- Augmented Reality
- Mixed Reality
- 3D Models

---



# Índice general

---

<b>1</b>	<b>Introducción</b>	<b>1</b>
1.1	Motivación . . . . .	1
1.2	Alcance y objetivos . . . . .	2
1.3	Estructura de la memoria . . . . .	2
<b>2</b>	<b>Estado del arte</b>	<b>5</b>
2.1	Conceptos previos . . . . .	5
2.1.1	Realidad aumentada, virtual y mixta . . . . .	5
2.1.2	Servidores y CPDs . . . . .	7
2.2	Trabajos relacionados . . . . .	8
<b>3</b>	<b>Tecnologías y herramientas</b>	<b>13</b>
3.1	Hardware . . . . .	13
3.1.1	Equipo . . . . .	13
3.1.2	HoloLens 2 . . . . .	14
3.2	Motor gráfico y bibliotecas . . . . .	15
3.2.1	Unity . . . . .	15
3.2.2	MRTK . . . . .	16
3.3	Monitorización de servidores . . . . .	17
3.3.1	Zabbix . . . . .	17
3.4	Desarrollo de código . . . . .	18
3.4.1	Visual Studio Code . . . . .	18
3.4.2	Visual Studio 2020 . . . . .	19
3.4.3	Git . . . . .	20
3.4.4	Sourcetree . . . . .	20
3.5	Representación gráfica . . . . .	20
3.5.1	Blender . . . . .	20
3.5.2	GIMP . . . . .	21

<b>4 Metodología, planificación y costes</b>	<b>23</b>
4.1 Metodología . . . . .	23
4.1.1 Scrum . . . . .	23
4.1.2 Kanban . . . . .	25
4.1.3 ScrumBan . . . . .	25
4.2 Planificación . . . . .	26
4.2.1 Sprints . . . . .	27
4.3 Diagrama de Gantt . . . . .	32
4.4 Estimación costes . . . . .	33
4.4.1 Recursos materiales . . . . .	33
4.4.2 Recursos software . . . . .	33
4.4.3 Recursos humanos . . . . .	33
4.4.4 Costes finales . . . . .	34
<b>5 Diseño funcional</b>	<b>35</b>
5.1 Introducción . . . . .	35
5.2 Pantalla de inicio . . . . .	36
5.2.1 Interfaz inicial . . . . .	36
5.2.2 Interfaz de opciones . . . . .	36
5.2.3 Interfaz de información . . . . .	38
5.3 Representación gráfica del entorno . . . . .	38
5.3.1 Lectura XMLs . . . . .	38
5.3.2 Modelos . . . . .	38
5.3.3 Elementos del entorno . . . . .	40
5.3.4 Funcionalidades de los modelos . . . . .	40
5.4 Representación de la información de Zabbix . . . . .	40
5.4.1 Información desde el modelo gráfico . . . . .	41
5.4.2 Interfaz principal . . . . .	41
5.4.3 Interfaz datos del servidor . . . . .	43
5.4.4 Interfaz de alertas . . . . .	43
5.4.5 Interfaz de datos de Zabbix . . . . .	44
5.4.6 Interfaz de máquinas virtuales . . . . .	44
5.4.7 Interfaz de ejecución de comandos . . . . .	45
5.5 User eXperience (UX) . . . . .	45
5.5.1 Menús auxiliares secundarios . . . . .	45
5.5.2 Funciones auxiliares . . . . .	46
5.5.3 Detección y muestra de errores . . . . .	49
5.5.4 Flujo de UX . . . . .	50

<b>6 Diseño técnico</b>	<b>53</b>
6.1 Introducción . . . . .	53
6.2 Metodología de la implementación . . . . .	53
6.2.1 Organización del proyecto de Unity . . . . .	53
6.2.2 Jerarquía de objetos de Unity . . . . .	57
6.2.3 Estándares de codificación y convenciones de nomenclatura del código	58
6.3 Pantalla de inicio . . . . .	59
6.3.1 Lectura y escritura del fichero de configuración . . . . .	60
6.3.2 Conexión con Zabbix . . . . .	60
6.3.3 Cifrado de credenciales . . . . .	61
6.4 Importación de los gráficos . . . . .	63
6.4.1 Modelos 3D . . . . .	63
6.4.2 Frontales de los servidores . . . . .	65
6.4.3 Controladores de los modelos 3D . . . . .	66
6.5 Carga de los gráficos mediante fichero XML . . . . .	68
6.6 Conexión de monitorización . . . . .	68
6.6.1 Conceptos de funcionamiento . . . . .	68
6.6.2 VPN . . . . .	69
6.6.3 VirManDC API . . . . .	70
6.6.4 Zabbix API . . . . .	70
6.6.5 JSONs de respuesta . . . . .	71
6.7 Representación de la información de Zabbix . . . . .	72
6.7.1 Información desde el modelo gráfico . . . . .	72
6.7.2 Elementos empleados de MRTK . . . . .	73
6.7.3 Interfaz principal . . . . .	76
6.7.4 Subinterfaces del servidor . . . . .	78
6.8 User eXperience (UX) . . . . .	78
6.8.1 Menús auxiliares secundarios . . . . .	78
6.8.2 Funciones auxiliares . . . . .	79
6.8.3 Detección y muestra de errores . . . . .	80
<b>7 Desarrollo y pruebas</b>	<b>81</b>
7.1 Desarrollo en PC y Hololens . . . . .	81
7.1.1 Entorno novedoso . . . . .	82
7.1.2 Entorno en movimiento . . . . .	82
7.1.3 Personalización plena y problemas asociados . . . . .	82
7.2 Pruebas . . . . .	83
7.2.1 Pruebas en funcionalidad . . . . .	83

7.2.2	Pruebas en fluidez . . . . .	83
7.2.3	Visual Profiler . . . . .	83
<b>8</b>	<b>Conclusiones y trabajo futuro</b>	<b>87</b>
8.1	Conclusiones . . . . .	87
8.1.1	Conclusiones generales . . . . .	87
8.1.2	Conclusiones personales . . . . .	88
8.2	Trabajo futuro . . . . .	89
<b>A</b>	<b>Guía de usuario</b>	<b>93</b>
A.1	Instalación del programa . . . . .	93
A.2	Configuración inicial . . . . .	96
A.3	Configuración ficheros XML . . . . .	96
A.3.1	Fichero Architecture_CITIC_CPD.xml . . . . .	97
A.3.2	Fichero Architecture_CITIC_CPD_little.xml . . . . .	99
A.3.3	Fichero IndicatorInterfacesModels.xml . . . . .	99
A.3.4	Fichero IndicatorsPanelModels.xml . . . . .	100
A.3.5	Fichero KeyValueModels.xml . . . . .	100
A.3.6	Fichero RackDataModels.xml . . . . .	101
A.3.7	Fichero ZabbixScripts.xml . . . . .	102
A.4	Configuración ficheros JSON . . . . .	102
A.4.1	Fichero Configuration.json . . . . .	103
A.4.2	Fichero DefaultConfiguration.json . . . . .	103
	<b>Lista de acrónimos</b>	<b>105</b>
	<b>Bibliografía</b>	<b>107</b>

# Índice de figuras

---

2.1	Fotografía de un usuario utilizando unas gafas de realidad virtual . . . . .	5
2.2	Captura de un teléfono inteligente empleando realidad aumentada . . . . .	6
2.3	Fotografía de un usuario utilizando unas gafas de realidad mixta . . . . .	6
2.4	Usuario de HoloLens 2 auditando los servidores de un CPD . . . . .	9
2.5	Captura de la aplicación de Taiwán en la que se aprecia una interfaz de usuario	9
2.6	Captura de la aplicación de Taiwán en la que se aprecia un modelo 3D . . . . .	10
2.7	Capturas de la aplicación Holoinventory en la que se aprecian interfaces y modelos 3D similares a VirManDC . . . . .	10
2.8	Captura del sistema holográfico de IAR . . . . .	11
3.1	HoloLens 2 . . . . .	14
3.2	Logotipo oficial de Unity . . . . .	15
3.3	Ejemplo de interfaz del editor Unity en el proyecto . . . . .	16
3.4	Logotipo de la librería MRTK . . . . .	16
3.5	Ejemplo de interfaz de Zabbix . . . . .	17
3.6	Ejemplo de interfaz de VS Code . . . . .	18
3.7	Ejemplo de interfaz de Visual Studio 2020 . . . . .	19
3.8	Logo oficial de Blender . . . . .	21
3.9	Logo oficial de GIMP . . . . .	21
4.1	Diagrama completo de las tareas del proyecto. Creado con GanttProject . . . . .	32
5.1	Captura de la pantalla de inicio . . . . .	35
5.2	Captura de la interfaz inicial . . . . .	36
5.3	Captura de la interfaz de opciones . . . . .	37
5.4	Captura de la subinterfaz de elección de arquitectura . . . . .	37
5.5	Captura de la interfaz de información . . . . .	38
5.6	Captura del modelo 3D de un armario rack . . . . .	39

5.7	Captura del modelo 3D de un servidor de rack . . . . .	39
5.8	Captura de la aplicación donde se aprecian dos servidores con alertas en su frontal . . . . .	41
5.9	Captura de la interfaz principal del servidor . . . . .	42
5.10	Captura de la interfaz de datos del servidor . . . . .	43
5.11	Captura de la interfaz de alertas del servidor . . . . .	44
5.12	Captura de la interfaz de alertas del servidor con una alerta activa . . . . .	44
5.13	Captura de la interfaz de datos de Zabbix del servidor . . . . .	45
5.14	Captura de la interfaz de la lista de VMs del servidor . . . . .	46
5.15	Captura de la interfaz principal de una máquina virtual . . . . .	47
5.16	Captura de la interfaz de comandos del servidor . . . . .	48
5.17	Captura del menú auxiliar de mano . . . . .	48
5.18	Captura de la interfaz del menú auxiliar principal . . . . .	49
5.19	Captura de la interfaz de información . . . . .	49
5.20	Captura de la interfaz de errores . . . . .	50
5.21	Diagrama de flujo del User eXperience . . . . .	51
6.1	Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta Assets . . . . .	54
6.2	Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta Resources . . . . .	54
6.3	Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta VirManDC . . . . .	55
6.4	Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta Scripts . . . . .	56
6.5	Captura de la interfaz de Unity en la que se aprecia la organización jerárquica del GameObject AdvancedModel_slot . . . . .	58
6.6	Código encargado de la encriptación de las credenciales . . . . .	62
6.7	Modelo 3D detallado del armario de rack APC NetShelter SX 42U . . . . .	63
6.8	Modelo 3D básico del armario de rack . . . . .	64
6.9	Modelo 3D intermedio del armario de rack . . . . .	64
6.10	Captura del contenido del GameObject BasicModel_Rack_LP . . . . .	66
6.11	Captura del contenido del GameObject AdvancedModel_Slot . . . . .	67
6.12	Captura de HoloLens 2 con botones de MRTK . . . . .	73
6.13	Captura de HoloLens 2 con el menú cercano . . . . .	74
6.14	Captura de HoloLens 2 con el menú de mano . . . . .	74
6.15	Captura de HoloLens 2 con control deslizante . . . . .	75
6.16	Captura de HoloLens 2 con el componente de información sobre herramientas	75

6.17 Captura de Unity en la que se aprecian los principales controladores de la interfaz principal . . . . .	76
7.1 Captura de la aplicación VirManDC en la que se aprecian los modelos 3D y la interfaz principal de servidor . . . . .	81
7.2 Captura de la herramienta Visual Profiler en la que se aprecian los recursos que monitoriza . . . . .	84
7.3 Captura de VirManDC en la que se aprecia el Visual Profiler . . . . .	84
7.4 Captura de VirManDC en la que se aprecia el Visual Profiler mostrando un elevado uso de recursos . . . . .	85
7.5 Captura de VirManDC con el Visual Profiler mostrando un pico en el uso de recursos . . . . .	85
A.1 Ajustes para la primera compilación del proyecto desde Unity . . . . .	94
A.2 Segunda compilación del proyecto desde Visual Studio, aparece el error mencionado . . . . .	94
A.3 Segunda compilación del proyecto desde Visual Studio, desaparece el error tras aplicar la solución . . . . .	95
A.4 Ruta del sistema de archivos de HoloLens en la que copiar la carpeta de configuración . . . . .	96
A.5 Captura de la interfaz web de las HoloLens 2, se muestra la ruta de instalación de los archivos de configuración . . . . .	97
A.6 Sección del fichero Architecture_CITIC_CPD.xml en la que se aprecia el atributo PlaceDescription . . . . .	97
A.7 Sección del fichero Architecture_CITIC_CPD.xml en la que se aprecia el atributo SlotsRack . . . . .	98
A.8 Sección del fichero Architecture_CITIC_CPD.xml en la que se aprecia la etiqueta Others . . . . .	99
A.9 Sección del fichero IndicatorInterfacesModels.xml . . . . .	99
A.10 Sección del fichero IndicatorsPanelModels.xml . . . . .	100
A.11 Sección del fichero KeyValueModels.xml . . . . .	101
A.12 Sección del fichero RackDataModels.xml . . . . .	101
A.13 Sección del fichero ZabbixScripts.xml . . . . .	102
A.14 Sección del fichero Configuration.json . . . . .	103
A.15 Sección del fichero DefaultConfiguration.json . . . . .	103



# Índice de tablas

---

4.1 Estimación de costes de los recursos humanos . . . . .	34
4.2 Estimación de costes de los recursos materiales . . . . .	34



# Capítulo 1

# Introducción

---

**E**n este primer capítulo se describe la motivación que ha llevado a la elaboración de este proyecto, el alcance y los objetivos que se pretenden lograr, y una breve descripción de la estructura de esta memoria.

## 1.1 Motivación

El correcto funcionamiento de un Centro de Procesamiento de Datos (CPD) requiere de una labor de supervisión constante por parte del personal técnico responsable para tratar de detectar y corregir lo antes posible cualquier incidencia que se pueda producir. Estas tareas de gestión de los activos del CPD conllevan una alta dedicación horaria y, en muchos casos, no pueden realizarse de forma remota. Ello implica el desplazamiento y revisión de la infraestructura en el propio CPD, para lo cual se requiere conocer detalladamente el plano con la posición de cada armario (*rack*) y de cada equipo [1]. Las características ambientales de un CPD, con unas condiciones de temperatura y humedad bajas para el funcionamiento óptimo de los equipos, y con unos niveles de ruido elevados debido al funcionamiento de los servidores y de los equipos de refrigeración [2], hacen que sea recomendable reducir lo máximo posible las estancias del personal técnico en su interior [3]. Además, en base a la pandemia COVID-19, ha quedado demostrado que es necesario disponer de un sistema alternativo a la modalidad presencial en el trabajo [4], por lo que una herramienta que imite lo máximo posible el entorno real, pero que pueda ser utilizada de forma remota, facilitará esa independencia al entorno de trabajo sin perder las ventajas del ámbito presencial.

A raíz de la experiencia al trabajar como encargado de sistemas de un CPD durante 2 años, concretamente en el CITIC<sup>1</sup>, surge la idea principal de este TFM: desarrollar una herramienta de visualización interactiva mediante realidad mixta para el apoyo a la gestión de las infraestructuras del CPD [5]. De esta forma, se tratarían de aplicar estas nuevas tecnologías para

---

<sup>1</sup> Centro de Investigación en Tecnologías de la Información y las Comunicaciones

ayudar al personal técnico en sus tareas diarias, mejorando los sistemas de monitorización existentes al dotarlos de una herramienta de visualización más natural e inmersiva.

Además, esta herramienta permitiría realizar auditorías de los recursos del CPD de forma remota, para reducir la posibilidad de riesgo médico y agilizar las tareas de mantenimiento. Por último, la aplicación no debería ser dependiente del CPD empleado como modelo (en este caso, el del CITIC), sino que debería ser capaz de adaptarse a cualquier otro modelo estructurado de armarios rack y servidores de cómputo.

## 1.2 Alcance y objetivos

El objetivo principal de este proyecto es el desarrollo de una herramienta de visualización inmersiva y específicamente diseñada para gafas de realidad mixta (HoloLens 2), que sirva de apoyo a las tareas de monitorización y supervisión de las infraestructuras de un CPD. Esta herramienta proporcionará al usuario información útil en tiempo real de los servidores tanto físicos como virtuales, y dará la posibilidad de interactuar con ellos de forma remota y mediante un canal seguro de comunicación. Además, se podrá intercambiar el modelo de CPD a representar sin ser necesario una reinstalación del programa.

Como objetivos secundarios que se pretenden conseguir con este proyecto se encuentran los siguientes:

- Mostrar y analizar el uso de entornos virtuales y/o holográficos para un caso de uso útil en el sector de las tecnologías de la información.
- Aplicar una tecnología puntera en cuyo ámbito apenas hay trabajo previo, y poder transmitir el conocimiento adquirido a otros posibles desarrolladores futuros.
- Poder utilizar este trabajo como sistema de soporte y formación para el aprendizaje del personal técnico en las tareas de monitorización de un CPD.
- Tener una alternativa no presencial para un entorno de trabajo en el que suele ser necesario estar presente.

## 1.3 Estructura de la memoria

Esta memoria estará conformada por los siguientes apartados:

- **Capítulo 1: Introducción**

En este capítulo se proporciona un contexto general al lector, se plantean los objetivos principales del proyecto y su motivación, y se explica la estructura de la memoria.

- **Capítulo 2: Estado del arte**

En este capítulo se explican diversos conceptos que se consideran de importancia para la correcta interpretación de la memoria, y se describe la búsqueda y revisión sobre el estado actual del campo al que pertenece el proyecto.

- **Capítulo 3: Tecnologías y herramientas**

En este capítulo se describen brevemente las tecnologías y herramientas empleadas para la elaboración del proyecto, además de las características especiales de ciertos componentes.

- **Capítulo 4: Metodología, planificación y costes**

En este capítulo se describen todos los pasos realizados para el desarrollo del proyecto, incluyendo la metodología empleada, las iteraciones en las que se dividió el diseño y la implementación de la herramienta.

- **Capítulo 5: Diseño funcional**

En este capítulo se describe el diseño final de las funcionalidades desarrolladas en la herramienta.

- **Capítulo 6: Diseño técnico**

En este capítulo se describe la implementación de las funcionalidades desarrolladas en la herramienta.

- **Capítulo 7: Desarrollo y pruebas**

En este capítulo se describe el desarrollo del proyecto, las características únicas del mismo y las diferentes pruebas realizadas para confirmar el correcto desarrollo de la aplicación.

- **Capítulo 8: Conclusiones y trabajo futuro**

En este capítulo se describen las conclusiones finales obtenidas al llegar al final del desarrollo del proyecto, y se proponen nuevos objetivos o funcionalidades que se podrían implementar en un futuro para ampliar las funcionalidades de la aplicación desarrollada.



## Capítulo 2

# Estado del arte

---

**E**n este capítulo se explican los conceptos previos necesarios para comprender el proyecto y un estudio del estado del arte acerca de trabajos relacionados con la solución propuesta.

## 2.1 Conceptos previos

### 2.1.1 Realidad aumentada, virtual y mixta

La Realidad Virtual (RV), la Realidad Aumentada (RA) y la Realidad Mixta (RM) son tecnologías inmersivas que han ganado popularidad en los últimos años [6]. Éstas ofrecen experiencias visuales y sensoriales que combinan elementos virtuales y físicos, pero difieren en la forma en que se presentan y se integran con el entorno del usuario.



Figura 2.1: Fotografía de un usuario utilizando unas gafas de realidad virtual

La Realidad Virtual (figura 2.1) se refiere a un entorno completamente generado por computadora en el que el usuario se sumerge por completo. A través de dispositivos como cascos o gafas especiales, se bloquea la percepción del mundo real y se sustituye por un mundo virtual. La RV utiliza técnicas de gráficos 3D, sonido y seguimiento de movimiento para crear una experiencia inmersiva en la que los usuarios pueden interactuar con objetos y entornos

virtuales de manera simulada. Esto se logra mediante la visualización de imágenes en este-reoscopía, que generan una sensación de profundidad y presencia.



Figura 2.2: Captura de un teléfono inteligente empleando realidad aumentada

Por otro lado, la **Realidad Aumentada** (figura 2.2) superpone información y elementos virtuales al entorno físico del usuario. A través de dispositivos como teléfonos inteligentes (smartphones) o gafas de RA, se captura y analiza el entorno real, y se añaden elementos virtuales como imágenes, videos o gráficos 3D sobre la vista del mundo real. La RA se basa en tecnologías como la detección de movimiento, reconocimiento de imágenes y seguimiento de posición para integrar los elementos virtuales de forma interactiva con el entorno físico.



Figura 2.3: Fotografía de un usuario utilizando unas gafas de realidad mixta

Por último, la **Realidad Mixta** o híbrida (figura 2.3) combina las dos anteriores, permitiendo no sólo aportar información virtual al entorno real, como la RA; sino que además es capaz de otorgar al usuario la capacidad de interactuar con ambas realidades, y que éstas respondan tanto a los estímulos del usuario como a los propios objetos tanto reales como virtuales.

Todas estas tecnologías tienen aplicaciones educativas significativas. Por ejemplo, la RV puede simular situaciones y entornos difíciles de acceder en la realidad, como viajar en el tiempo, explorar el espacio o sumergirse en ambientes históricos.

La RA, por su parte, puede enriquecer la educación al superponer información adicional sobre objetos del mundo real. Por ejemplo, los estudiantes pueden escanear una obra de arte con una aplicación de RA y obtener información detallada sobre la pintura, su autor y su contexto histórico.

Para finalizar, la capacidad interactiva de la RM permite un aprendizaje avanzado de la información mostrada sobre el mundo real, al ser capaz de reaccionar a las acciones en tiempo real del usuario.

### 2.1.2 Servidores y CPDs

En el contexto de este proyecto, un servidor es un sistema físico o virtual diseñado para proporcionar servicios, recursos o datos a otros sistemas o programas, conocidos como *clientes*. Los servidores son componentes esenciales en arquitecturas de red y sistemas distribuidos, ya que gestionan y controlan el acceso y la entrega de información a los clientes que lo solicitan.

Los servidores pueden ofrecer una variedad de servicios, como:

- Servidores web: Entregan páginas web y otros recursos a través de internet utilizando protocolos como HTTP.
- Servidores de cómputo: Optimizados para el cálculo avanzado, ofrecen características muy potentes a clientes para herramientas de IA o ML, entre otros.
- Servidores de bases de datos: Almacenan, gestionan y permiten el acceso a bases de datos, proporcionando consultas y almacenamiento de información.
- Servidores de archivos: Comparten y almacenan archivos en una red, permitiendo a los clientes acceder a esos archivos de manera centralizada.
- Servidores de aplicaciones: Ejecutan y gestionan aplicaciones y servicios que pueden ser accedidos por múltiples usuarios.

Un **CPD**, o **Centro de Procesamiento de Datos** (también conocido como Data Center en inglés), es una instalación física que alberga una gran cantidad de servidores y equipos relacionados necesarios para el almacenamiento, procesamiento y distribución de datos y aplicaciones. Los **CPD** son esenciales para la operación de servicios en línea, infraestructuras de tecnología de la información y servicios en la nube.

Un CPD típicamente incluye:

- Servidores: Equipos que proporcionan los servicios y recursos a los clientes.
- Redes: Infraestructura de red que permite la comunicación entre servidores y clientes, y la conexión a internet.
- Sistemas de almacenamiento: Dispositivos de almacenamiento como unidades de disco duro, matrices de almacenamiento o sistemas de almacenamiento en red ([NAS](#)) para guardar datos.
- Sistemas de refrigeración y energía: Sistemas para mantener la temperatura adecuada y sistemas de energía para asegurar el funcionamiento continuo de los equipos, como los [SAI](#)<sup>1</sup>.
- Medidas de seguridad: Sistemas de seguridad física y lógica para proteger los datos y equipos contra amenazas.
- Gestión y monitorización: Herramientas y sistemas para administrar, monitorizar, supervisar y mantener los equipos y servicios en el CPD.

En resumen, un CPD es el lugar físico donde se alojan y operan servidores y equipos relacionados para garantizar la disponibilidad y entrega de servicios digitales.

## 2.2 Trabajos relacionados

Existen aplicaciones similares a este proyecto, aunque recientes y con escasa publicidad y trayectoria. Un ejemplo de esto sería el uso de HoloLens 2 por parte de un CPD europeo en Polonia a partir del año 2021 [7], pero en este caso como herramienta de apoyo visual para las tareas de mantenimiento requeridas por los propietarios de los servidores y realizadas por los responsables del centro (figura 2.4). Gracias a las capacidades de las HoloLens 2, los clientes pueden comunicarse y supervisar las tareas de [SOP](#)<sup>2</sup> por parte de los trabajadores del CPD sin que sea necesario tener que situarse presencialmente en el centro.

---

<sup>1</sup> Sistema de Alimentación Ininterrumpida que proporciona tensión a un conjunto de equipos durante un corto periodo de tiempo para que pueda comutar el grupo eléctrico en caso de fallo de tensión.

<sup>2</sup> Un [Standard Operating Procedure \(SOP\)](#) es un proceso estándar para realizar determinada acción, como puede ser el mantenimiento de un servidor.



Figura 2.4: Usuario de HoloLens 2 auditando los servidores de un CPD

También Microsoft ha realizado pruebas similares en 2021 [8] en sus propios servidores, demostrando las capacidades de la herramienta para monitorizar y auditar diversos procesos de un CPD.

La herramienta más similar a la que se plantea en este proyecto es la encontrada en un vídeo de Youtube [9] del año 2022 (figura 2.5), en la que aparentemente un grupo universitario de Taiwán muestra la demo de una aplicación de monitorización y ayuda a SOPs de un CPD empleando HoloLens 2.



Figura 2.5: Captura de la aplicación de Taiwán en la que se aprecia una interfaz de usuario

Al contrario que VirManDC, esta aplicación parece mostrar los datos mediante el navegador web, y ampliando los datos según los parámetros ambientales, pero sin mostrar el contenido de los servidores.

También emplea Zabbix como herramienta de apoyo a la monitorización y representa modelos 3D (figura 2.6), pero es necesario situarse en el interior del CPD pues requiere de componentes como códigos QR para establecer un enlace. Este proyecto se desarrolló de forma totalmente paralela y sin conexión alguna con la herramienta planteada en este trabajo.



Figura 2.6: Captura de la aplicación de Taiwán en la que se aprecia un modelo 3D

Por último, la empresa croata Inceptum ofrece desde el año 2021 un producto privativo llamado Holoinventory [10] con características similares a la herramienta propuesta para este proyecto, pero sin suponer una alternativa remota ya que está orientada a las tareas de mantenimiento de forma presencial (figura 2.7). A diferencia de VirManDC, este producto emplea diferentes aplicaciones privadas con licencias de pago, lo que conlleva un sobrecoste elevado al producto final.

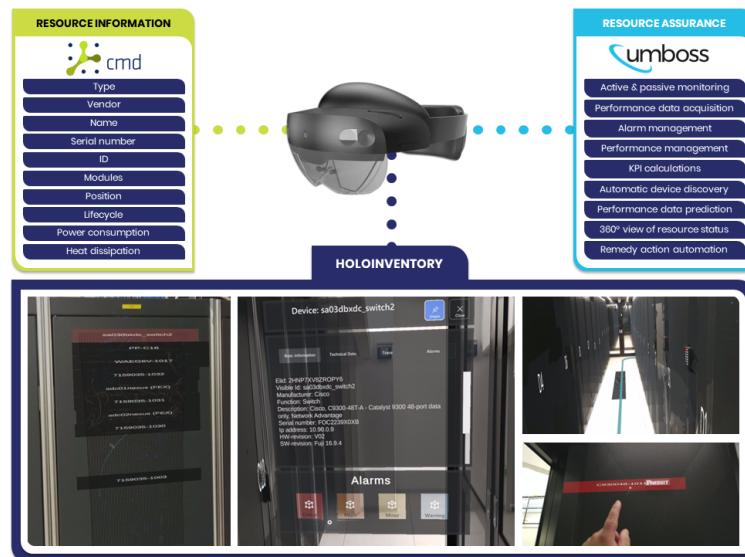


Figura 2.7: Capturas de la aplicación Holoinventory en la que se aprecian interfaces y modelos 3D similares a VirManDC

Fuera del campo de supervisión de CPDs, también encontramos otros usos de las HoloLens 2, incluyendo en la ingeniería [11] y la robótica [12], para visualizar información en tiempo real y mejorar la eficiencia del trabajo. Concretamente en el entorno médico, ha habido muchos avances en los últimos años en el campo de la realidad aumentada [13], y los expertos empiezan a utilizar de forma asidua estas nuevas herramientas. No está libre de problemas [14], como la dependencia de las baterías o el común mareo tras uso intensivo de plataformas de realidad virtual o aumentada; pero existen cada vez más soluciones de esta naturaleza [15] y lentamente se van integrando en el ecosistema de la medicina.

Como inspiración para este proyecto podemos citar el *TFG Sistema de ayuda a la supervisión de infraestructuras informáticas mediante técnicas de visualización inmersivas. Integración con una herramienta de monitorización de código abierto* [16]; del cual se tomaron como modelo para este proyecto conceptos básicos como la interacción entre interfaces-usuarios y obtención de datos mediante herramientas de código abierto. Sin embargo, estas ideas están aplicadas sobre tecnologías e interfaces comunes, sin tener en cuenta su posible aplicación en entornos de realidad aumentada o mixta.

Por otro lado, el proyecto *A Collaborative Industrial Augmented Reality Digital Twin: Developing the Future of Shipyard 4.0* [17] describe las capacidades de las herramientas holográficas para mejorar las condiciones de aprendizaje de los trabajadores industriales a la vez que se reducen los costes y riesgos de dicha enseñanza. Incorpora el concepto de IAR como base para el futuro paradigma de Industria 4.0, sirviendo de inspiración para implementar una herramienta similar enfocada en las tareas de un administrador de sistemas.

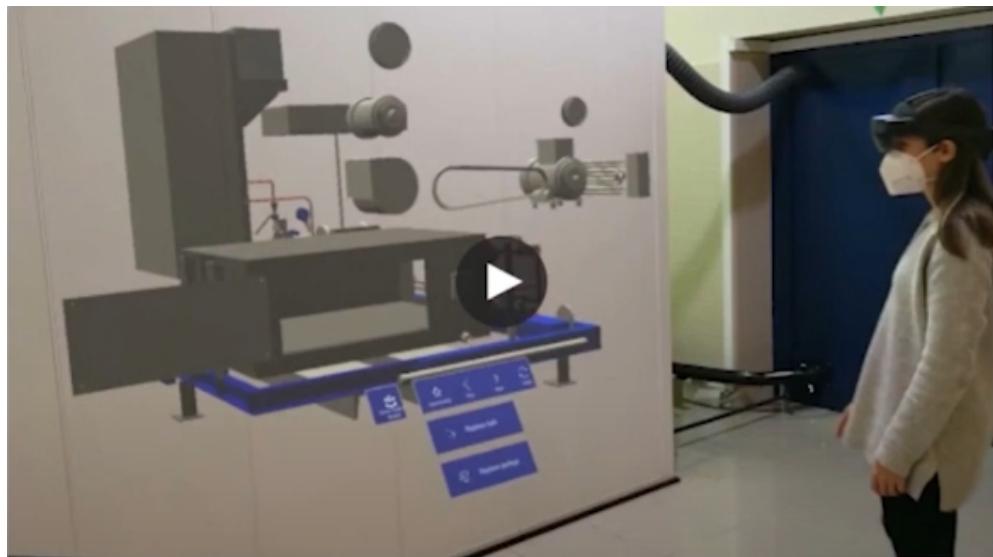


Figura 2.8: Captura del sistema holográfico de IAR



## Capítulo 3

# Tecnologías y herramientas

---

**E**n este capítulo se ofrece una contextualización de las herramientas y tecnologías empleadas en el mismo.

## 3.1 Hardware

En este apartado se describen las características principales del hardware empleado, distinguiendo entre el equipo y las HoloLens 2.

### 3.1.1 Equipo

El equipo empleado para todo el proceso (desarrollo, implementación y pruebas) ha sido un Asus con las siguientes características:

- Modelo Asus Rog Strix G15 G512LV
- CPU Intel(R) Core(TM) i7-10750H CPU @ 2.60GHz 2.59 GHz
- Memoria 16GB DDR4
- Tarjeta gráfica NVIDIA GeForce RTX 2060

Como se puede observar, se trata de un dispositivo bastante potente en cuanto a capacidades gráficas y de cómputo. Esto se debe al gran consumo de recursos que requiere la replicación e implementación del apartado gráfico de la realidad mixta. De haber empleado un PC con características más básicas, la implementación y depuración del proyecto hubiese sido más tediosa, si no incluso inviable.

### 3.1.2 HoloLens 2



Figura 3.1: HoloLens 2

Las HoloLens 2 [18] son un dispositivo de Realidad Mixta (RM) desarrollado y fabricado por Microsoft (figura 3.1). Son unas gafas inteligentes que permiten interactuar con objetos virtuales superpuestos en el mundo real y, a diferencia de la Realidad Virtual (RV), donde se crea un entorno completamente virtual, la realidad aumentada combina elementos virtuales con el entorno real que rodea al usuario.

Las características más importantes de las HoloLens 2 son:

- Sensores y cámaras integradas (luz visible, infrarrojos)
- Sistema de seguimiento ocular
- Conjunto de micrófonos y altavoces
- Lentes holográficas transparentes (guías de ondas)
- Control de gestos y reconocimiento de voz
- Acelerómetro, giroscopio, magnetómetro
- Seguimiento posicional a escala mundial
- Malla de entorno en tiempo real

En cuanto a las especificaciones, son las siguientes:

- CPU Qualcomm Snapdragon 850
- Memoria DRAM LPDDR4x DE 4 GB
- Almacenamiento UFS 2.1 de 64 GB
- Wi-Fi 5 (802.11ac 2x2)

Todas estas características permiten a las HoloLens 2 superponer hologramas en 3D en el mundo real, ofreciendo una experiencia interactiva e inmersiva. Se seleccionó esta herramienta para el desarrollo de este proyecto debido a que el CITIC disponía inicialmente de ellas, y dentro de los aparatos holográficos, es el de mayor comodidad para el usuario. Además, dispone de diversas librerías y recursos de código abierto para el desarrollo de aplicaciones específicas para esta herramienta.

## 3.2 Motor gráfico y librerías

En este apartado se cita el motor gráfico empleado para la creación del sistema virtual, y las librerías de importancia asociadas a éste.

### 3.2.1 Unity



Figura 3.2: Logotipo oficial de Unity

El motor gráfico Unity [19] ha sido empleado desde hace muchos años (su primera versión data del año 2005) en todo tipo de proyectos que requieran de un potente motor gráfico 3D capaz de computar simulaciones físicas con rapidez. Destaca por ser uno de los más potentes y versátiles, y permite su uso gratuito a través de las licencias *Unity Personal*, estando disponible para los sistemas operativos Windows, Mac OS y GNU/Linux. Su motor gráfico permite el renderizado mediante OpenGL para todas las plataformas o Direct3D exclusivamente en plataformas Windows, mientras que el scripting se realiza a través de Boo, Mono o C#.

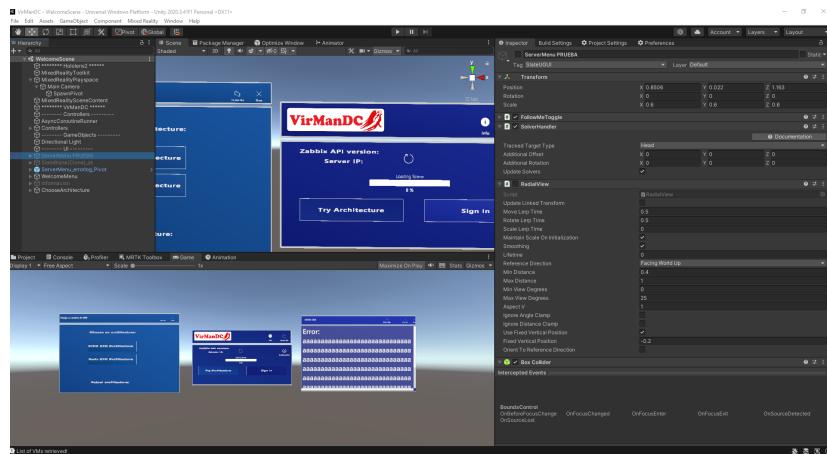


Figura 3.3: Ejemplo de interfaz del editor Unity en el proyecto

Se ha seleccionado este motor gráfico para este proyecto ya que cuenta con un soporte nativo para la plataforma de HoloLens 2, lo que significa que se pueden crear aplicaciones que se ejecutan directamente en el dispositivo sin la necesidad de realizar ajustes significativos. Además, Unity cuenta con una gran cantidad de bibliotecas y herramientas de desarrollo para la creación de aplicaciones en 3D, lo que facilita el proceso de desarrollo de aplicaciones en HoloLens 2.

Como alternativa de motor gráfico diseñado para este tipo de proyectos y compatible con las HoloLens 2 se encuentra Unreal Engine [20], pero rechazado para este proyecto debido a que el equipo técnico no tiene experiencia con el mismo.

### 3.2.2 MRTK



Figura 3.4: Logotipo de la librería MRTK

MRTK (Mixed Reality ToolKit) [21] es un conjunto de herramientas de software de código abierto desarrollado por Microsoft y la comunidad de desarrolladores de realidad mixta que se utiliza para crear aplicaciones orientadas a plataformas como HoloLens, Windows Mixed Reality y dispositivos compatibles con OpenXR.

MRTK proporciona una variedad de componentes y sistemas de interfaz de usuario que permiten a los desarrolladores crear aplicaciones de realidad mixta de manera más rápida y eficiente. Estos componentes incluyen objetos interactivos como botones, menús y paneles, así como sistemas de entrada y salida de usuario, seguimiento de movimiento, reconocimiento de voz y gestos, entre otros.

Estas herramientas también ofrecen una arquitectura flexible y modular que permite a los desarrolladores personalizar y ampliar el conjunto de herramientas según sus necesidades y requerimientos específicos de la aplicación. Además, MRTK se integra perfectamente con Unity, lo que facilita el proceso de desarrollo para los desarrolladores que ya tienen experiencia con esta plataforma.

En resumen, MRTK es una herramienta esencial para los desarrolladores de aplicaciones de realidad mixta que buscan crear experiencias de usuario de alta calidad y eficientes en plataformas como HoloLens y Windows Mixed Reality.

### 3.3 Monitorización de servidores

En este apartado se describe la herramienta seleccionada para la monitorización de los servidores del CPD.

#### 3.3.1 Zabbix

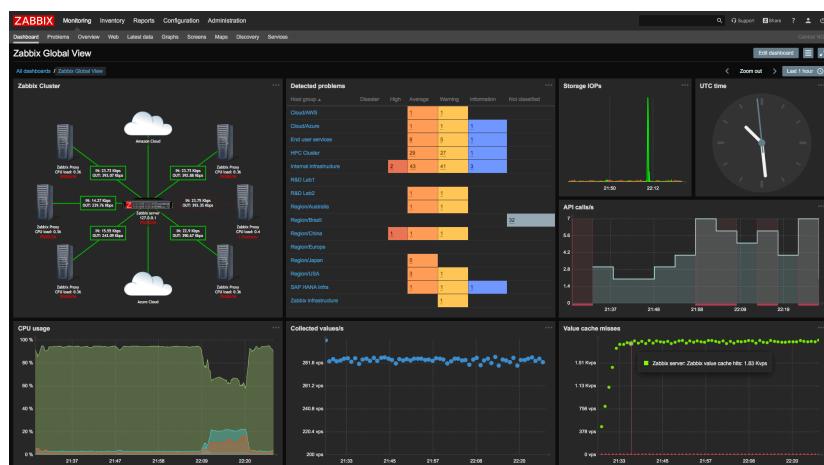


Figura 3.5: Ejemplo de interfaz de Zabbix

Zabbix [22] es una plataforma de monitorización de código abierto diseñada para supervisar el rendimiento y la disponibilidad de sistemas, redes y servicios en tiempo real. Permite a los administradores de sistemas y redes recopilar datos, detectar problemas y tomar medidas preventivas para garantizar un funcionamiento óptimo de los activos tecnológicos. Zabbix ofrece características como la supervisión de métricas, alertas configurables, visualización de datos en tableros y gráficos, y generación de informes. Se destaca por su capacidad de escalabilidad y su flexibilidad para adaptarse a diversas infraestructuras y entornos.

Ofrece una *API* de servicio con la que se puede interactuar para generar consultas o editar parámetros, por lo que es la plataforma ideal para desarrollar un proyecto similar al aquí expuesto.

Aunque existen otras alternativas de código abierto de monitorización de servidores como son Nagios [23] o Zennos [24], en este proyecto se empleó Zabbix al ser una de las aplicaciones más utilizadas en el sector. Además, el propio CPD del CITIC cuenta con esta herramienta.

## 3.4 Desarrollo de código

En este apartado se citan las herramientas empleadas para el desarrollo del código de la aplicación, y los sistemas de control de versiones.

### 3.4.1 Visual Studio Code

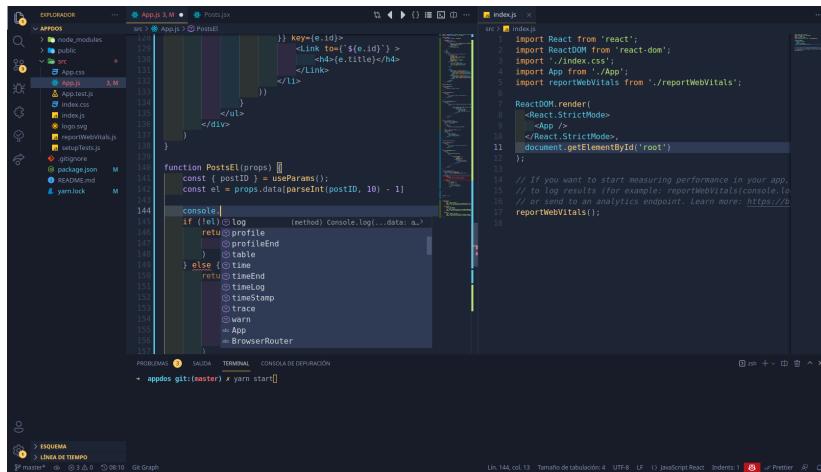


Figura 3.6: Ejemplo de interfaz de VS Code

Visual Studio Code (VS Code) [25] es un editor de código fuente desarrollado por Microsoft. Se destaca por ser gratuito, ligero y altamente personalizable, lo que lo convierte en una de las herramientas más populares para programadores y desarrolladores. VS Code ofrece soporte para una amplia gama de lenguajes de programación, proporcionando resaltado de

sintaxis, autocompletado inteligente, depuración integrada y muchas extensiones que añaden funcionalidades adicionales. Su interfaz intuitiva y su integración con control de versiones, como Git, lo convierten en una elección versátil tanto para proyectos pequeños como para proyectos más complejos.

Dispone, entre muchas otros, de extensiones centradas en Unity, C# o .Net, por lo que la mayor parte del código del proyecto ha sido desarrollado en esta plataforma.

### 3.4.2 Visual Studio 2020

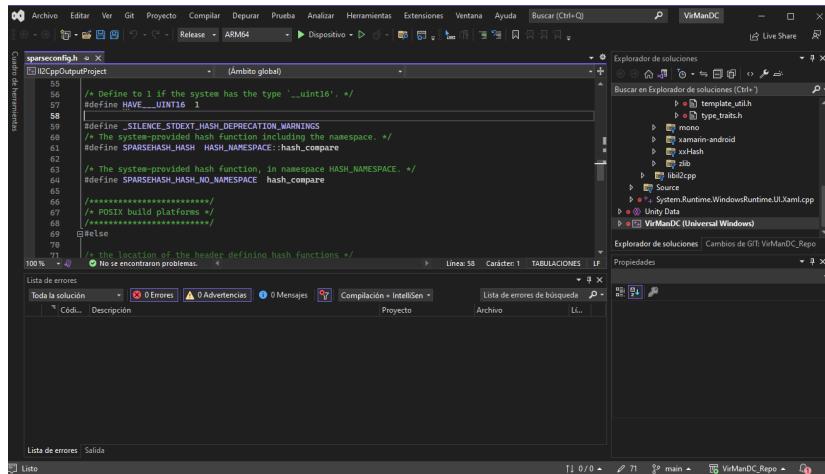


Figura 3.7: Ejemplo de interfaz de Visual Studio 2020

Microsoft Visual Studio [26] es un entorno de desarrollo integrado (IDE) para Windows y macOS. Es compatible con múltiples lenguajes de programación, tales como C++, C#, Visual Basic .NET, F#, Java, Python, Ruby y PHP, al igual que entornos de desarrollo web, como ASP.NET MVC o Django, a lo cual hay que sumarle las nuevas capacidades en línea bajo Windows Azure en forma del editor Mónaco.

Visual Studio permite a los desarrolladores crear sitios y aplicaciones web, así como servicios web en cualquier entorno compatible con la plataforma .NET. Así, se pueden crear aplicaciones que se comuniquen entre estaciones de trabajo, páginas web, dispositivos móviles, dispositivos embebidos y videoconsolas, entre otros.

A diferencia de VS Code, que es únicamente un editor de código, Visual Studio permite la compilación y depuración de proyectos. Este software es necesario para instalar la herramienta en el dispositivo HoloLens, tal y como se explica en detalle en el anexo A.1.

### 3.4.3 Git

Git [27] es un sistema de control de versiones distribuido ampliamente utilizado en el desarrollo de software. Permite a los desarrolladores rastrear y administrar cambios en el código fuente y colaborar en proyectos de manera eficiente. Dado que almacena el historial completo de cambios, la creación de ramas para trabajar en nuevas características o solucionar problemas es un proceso sencillo, sin afectar a la rama principal. Además, la capacidad de fusionar y resolver conflictos de manera efectiva es una característica clave. Git es esencial para el desarrollo colaborativo y la gestión de proyectos de software.

Al ser una herramienta tan popular, las demás herramientas presentan características optimizadas para la utilización avanzada de Git. En este proyecto, al ser un equipo técnico reducido, se limita su uso a copia de seguridad del proyecto.

### 3.4.4 Sourcetree

Sourcetree [28] es una interfaz gráfica de usuario (GUI) para el sistema de control de versiones Git. Desarrollado por Atlassian, Sourcetree simplifica la gestión de repositorios Git al proporcionar una forma visual de realizar tareas como clonar repositorios, crear ramas, fusionar cambios y gestionar conflictos. Es especialmente útil para aquellos que prefieren una interfaz gráfica en lugar de la línea de comandos para trabajar con Git. Sourcetree está disponible para sistemas operativos Windows y macOS, y ofrece una manera intuitiva de interactuar con repositorios Git y colaborar en proyectos de desarrollo de software.

Si bien las operaciones con Git en este proyecto son sencillas, el empleo de esta herramienta dota de comodidad al uso de control de versiones.

## 3.5 Representación gráfica

En este apartado se describen brevemente las aplicaciones utilizadas para la implementación de los gráficos del proyecto.

### 3.5.1 Blender

Blender [29] es un software de modelado, animación y renderización en 3D de código abierto. Es utilizado por artistas, diseñadores y animadores para crear contenido visual, desde películas y videojuegos hasta animaciones y efectos especiales. Esta plataforma ofrece una amplia gama de herramientas para modelar objetos en 3D, aplicar texturas, animar personajes y objetos, simular física y renderizar imágenes y animaciones finales.

Es compatible con Unity ya que existen diversas facilidades para emplear los modelos 3D obtenidos en Blender en dicha herramienta. En este proyecto se emplea para preparar y corregir los modelos 3D utilizados en la representación gráfica.



Figura 3.8: Logo oficial de Blender

Puesto que es de licencia libre, una de las herramientas de modelización más empleadas del sector y el equipo técnico cuenta con experiencia con la aplicación, se decidió el empleo de esta herramienta sobre otras alternativas como FreeCAD [30], también de código abierto.

### 3.5.2 GIMP



Figura 3.9: Logo oficial de GIMP

GIMP (GNU Image Manipulation Program) [31] es un programa de edición de imágenes de código abierto y gratuito. Ofrece herramientas y funciones para retocar fotos, crear gráficos, manipular imágenes y realizar ediciones creativas. Esta herramienta supone una alternativa popular a software comercial como Adobe Photoshop y brinda características como capas, filtros, herramientas de selección, pinceles y soporte para múltiples formatos de imagen. Es utilizado por diseñadores gráficos, fotógrafos y artistas para realizar tareas de edición y manipulación de imágenes. En este proyecto se emplea principalmente para la modificación de ciertas texturas de los modelos gráficos, y para conseguir un correcto formato de los frontales de los servidores.



## Capítulo 4

# Metodología, planificación y costes

---

**E**n este capítulo se expone la metodología elegida para la realización del proyecto. Se define la metodología ágil *ScrumBan* y se describe la planificación basada en objetivos. Adicionalmente se definen los sprint del proyecto y, por último, se presenta un resumen de los costes totales estimados.

## 4.1 Metodología

Debido a la naturaleza de este proyecto, se ha decidido basarse en una metodología ágil de trabajo [32], ya que el desarrollo de esta herramienta requiere flexibilidad, adaptabilidad y una respuesta rápida a los cambios y necesidades del proyecto. Puesto que se trata de una herramienta que carece de proyectos similares en los que basarse, es posible que los requerimientos cambien en espacios cortos de tiempo, y puesto que las interacciones deben ser frecuentes (presentes en este proyecto como prototipos a mostrar en reuniones mensuales con los directores, que actúan como clientes) requieren de entregas continuas y progresivas.

Concretamente, se ha decidido basarse en la metodología **ScrumBan**, que combina **Scrum** y **Kanban**, dos de las metodologías más utilizadas en el desarrollo ágil de proyectos.

### 4.1.1 Scrum

Scrum [33] es un marco de trabajo ágil ampliamente utilizado para la gestión y desarrollo de proyectos, especialmente en el ámbito del desarrollo de software, basado en la colaboración, la adaptación y la entrega incremental de valor. Como elemento principal, los entregables son ligeros y muy concretos, de esta forma se agiliza el proceso de desarrollo y se tienen versiones funcionales del producto en cada iteración. Además, ofrece la flexibilidad necesaria para adaptarse a nuevos problemas, como son posibles cambios en los requisitos, un nuevo método de enfoque al problema o posibles retrasos en alguna tarea concreta.

Según la *guía oficial de Scrum*[34], este marco de desarrollo define los siguientes elementos esenciales:

- *Scrum Team*
  - *Developers*: Encargados de crear cualquier aspecto de un incremento funcional en cada *Sprint*.
  - *Product Owner*: Encargado de maximizar el valor del producto obtenido como resultado del trabajo del *Scrum Team*.
  - *Scrum Master*: Encargado de que se siga la metodología de Scrum tanto por parte del *Scrum Team* como por la organización.
- *Scrum Events*
  - *Sprint*: Eventos de duración fija de un mes o menos durante los que se realiza el trabajo.
  - *Sprint Planning*: Este evento inicia el *Sprint* definiendo el trabajo a realizar.
  - *Daily Scrum*: Reunión diaria de 15 minutos en la que se inspecciona el progreso realizado hasta ese momento y se reajusta de ser necesario el trabajo futuro.
  - *Sprint Review*: Evento en el que se analiza el *Sprint* al momento de su finalización y se determinan futuras adaptaciones.
  - *Sprint Retrospective*: En este evento se estudia el último *Sprint* para buscar formas de incrementar la calidad y la efectividad, debatiendo sobre problemas encontrados y las soluciones aplicadas.
- *Scrum Artifacts*
  - *Product Backlog*: Lista ordenada y cambiante que contiene lo necesario para mejorar el producto.
  - *Sprint Backlog*: Se compone de *Sprint Goal*, la lista de elementos del *Product Backlog* elegidos para el *Sprint* y el plan de acción que define cómo se entregará el *Increment*.
  - *Increment*: Representa un hito en forma de entregable hacia la finalización del producto.

Dentro de la propia guía se indica que el *Scrum Team* debe ser lo suficientemente pequeño para continuar siendo ágil y lo suficientemente grande para completar una cantidad significativa de trabajo en un *Sprint*. Teniendo en cuenta esta definición y que en un trabajo de fin de máster es el estudiante quien realiza el trabajo orientado por sus directores, no se puede

aplicar esta metodología de forma estricta cuando el *Scrum Team* está limitado a una persona y sus supervisores. Sin embargo, sí tiene interés hacer uso de otros elementos de Scrum como son los *Sprints* o el *Product Backlog*, ya que aportan valor al proyecto que se va a desarrollar.

#### 4.1.2 Kanban

Kanban [35] es un enfoque de gestión visual utilizado para optimizar la eficiencia y la agilidad en los procesos de trabajo. Originalmente desarrollado en Toyota en la industria manufacturera [36], ha sido adaptado y adoptado en diversas áreas, incluyendo el desarrollo de software y la gestión de proyectos. Esta metodología define las siguientes características:

- *Tablero Kanban*: Se utiliza un tablero visual para representar el flujo de trabajo. El tablero suele estar dividido en columnas que representan etapas o estados del proceso, desde la entrada de tareas hasta su finalización.
- *Tarjetas*: Cada tarea, elemento o trabajo se representa mediante tarjetas o notas adhesivas en el tablero. Estas tarjetas representan unidades de trabajo individuales.
- *Columnas*: Las columnas en el tablero representan los diferentes estados o fases por los que pasan las tareas. Suelen reducirse a "Por Hacer", "En Progreso" y "Hecho".
- *Límites de WIP*: WIP significa "Work in Progress" (Trabajo en Progreso). Kanban limita la cantidad de trabajo que puede estar en proceso en una columna específica. Esto ayuda a prevenir el exceso de trabajo en curso, lo que puede causar cuellos de botella y retrasos.
- *Flujo Continuo*: Kanban se enfoca en mantener un flujo constante de trabajo. A medida que se completan tareas en una columna, se mueven a la siguiente. Esto minimiza el tiempo de espera y mejora la eficiencia.

En resumen, Kanban es un enfoque visual que optimiza el flujo de trabajo, fomenta la eficiencia y mejora continua, y se adapta a una variedad de contextos. Es especialmente útil cuando se busca una gestión más ágil y flexible de las tareas y proyectos.

#### 4.1.3 ScrumBan

ScrumBan [37] es una metodología híbrida que integra elementos de Scrum y Kanban para gestionar proyectos y flujos de trabajo. Combina la estructura y los eventos de Scrum con la flexibilidad y el enfoque en el flujo continuo de Kanban.

Los equipos que adoptan ScrumBan generalmente mantienen ciertos aspectos de Scrum, como los roles de Scrum (Product Owner, Scrum Master, Equipo de Desarrollo) y algunos de los eventos (Sprint Planning, Daily Scrum, Sprint Review, Sprint Retrospective). Sin embargo, en lugar de trabajar en sprints fijos, ScrumBan permite que las tareas fluyan de manera

más continua, similar al enfoque Kanban. Esto se ve representado en la diferencia entre la planificación y los sprints desarrollados finalmente, que aunque similares, no siguen la misma estructura, pues se prioriza la optimización de las tareas según van cambiando los requisitos del proyecto.

## 4.2 Planificación

Tomando como base los objetivos principales definidos en la sección 1.2, se ha planteado una planificación que permita cumplir dichos objetivos de manera ordenada y por etapas diferenciadas. Cada una de estas etapas o fases se descompone en actividades que será necesario completar para dar por finalizada dicha fase. Algunas actividades dentro de una fase podrán ser realizadas en paralelo, al no depender de otras anteriores, como se puede ver en el diagrama de Gantt con la planificación inicial (Sección 4.3). Para poder continuar con la siguiente fase, es necesario avanzar lo suficiente en la anterior, pero al ser tan complejas no se considera imprescindible completar los requisitos en su totalidad, pudiendo volver en sprints posteriores a una fase anterior.

1. Fase 1. Sistema de representación gráfica del CPD
  - (a) Obtención de modelos 3D
  - (b) Diseño de carga dinámica
  - (c) Implementación del sistema en entorno de desarrollo
  - (d) Instalación y pruebas en plataforma HoloLens2
2. Fase 2. Conexión al servidor Zabbix
  - (a) Diseño e implementación de conexión
  - (b) Obtención datos básicos
  - (c) Obtención datos avanzados
  - (d) Instalación y pruebas en plataforma HoloLens2
3. Fase 3. Diseño de interfaces del usuario
  - (a) Diseño de interfaz principal del servidor
  - (b) Diseño interfaces secundarias del servidor
  - (c) Diseño interfaces auxiliares
4. Fase 4. Implementación de interfaces con MRTK

- (a) Implementación de interfaz principal del servidor
  - (b) Implementación interfaces secundarias del servidor
  - (c) Implementación interfaces auxiliares
  - (d) Instalación y pruebas en plataforma HoloLens2
5. Fase 5. Separación del proyecto en escenas
    - (a) Escena de inicio aplicación
    - (b) Escena principal
    - (c) Instalación y pruebas en plataforma HoloLens2
  6. Fase 6. Diseño de funciones extra
    - (a) Diseño de extras de escena de inicio aplicación
    - (b) Diseño de extras de escena principal
  7. Fase 7. Implementación de funciones extra
    - (a) Implementación de extras de escena de inicio aplicación
    - (b) Implementación de extras de escena principal
    - (c) Instalación y pruebas en plataforma HoloLens2

#### 4.2.1 Sprints

En base a la planificación planteada en la sección 4.2 se definieron los siguientes *sprints*, que tendrán una duración de **dos semanas** por defecto, aunque se puede observar la duración final en el diagrama de Gantt de la sección 4.3.

##### Sprint 1. Obtención de modelos 3D y diseño de carga dinámica

Tras una búsqueda en Internet, se obtuvieron los principales modelos 3D del proyecto, consistentes en los armarios de rack. Además, se intentaron extraer los frontales de los servidores mediante fotos editadas. Para cargar dichos modelos se diseñó un documento XML básico que pudiese ser leído por la plataforma. Ambos procesos se categorizan como parte de las tareas de la fase 1, concretamente la 1 y la 2.

*Asignación de horas por recurso:*

- Analista: 10h
- Diseñador Software: 20h
- Programador: 50h

### Sprint 2. Implementación de sistema de lectura de ficheros XML

Se trabajó en implementar el sistema de lectura de ficheros XML, tal y como se describe en la sección [6.5](#) (aunque de forma más básica).

*Asignación de horas por recurso:*

- Programador: 80h

### Sprint 3. Instalación y pruebas en las HoloLens2

En base a la tarea [1d](#), se realizaron la instalación y pruebas de la representación gráfica. Dado que fueron necesarios varios ajustes debido a la lectura de ficheros externos a la aplicación y al escaso conocimiento de la plataforma, este sprint se retrasó una semana.

*Asignación de horas por recurso:*

- Programador: 80h
- Tester: 40h

### Hito fase 1

Una vez finalizado el sprint 3, se obtuvo un [Producto Mínimo Viable \(MVP\)](#)<sup>1</sup> que fue aceptado por los directores. Si bien se trataba de una versión sencilla y no libre de errores, ofrecía las características mínimas requeridas para este desarrollo en cuanto a la representación gráfica del entorno.

### Sprint 4. Diseño de conexión y obtención de datos básicos

En base a la tarea [2a](#), se realizó el diseño e implementación de una conexión a la [API](#) de Zabbix. Posteriormente se obtuvieron los primeros datos básicos de los servidores monitORIZados, cumpliendo así la tarea [2b](#). Debido a la complejidad de la implementación de las conexiones usando las librerías de C# de Unity, esta tarea se retrasó una semana.

*Asignación de horas por recurso:*

- Diseñador Software: 60h
- Programador: 60h

---

<sup>1</sup> En software, el Producto Mínimo Viable (MVP, por sus siglas en inglés) es una versión mínima de un producto nuevo que incluye las características básicas para satisfacer las necesidades de los clientes

### Sprint 5. Obtención de datos avanzados e instalación y pruebas

Una vez implementada la conexión, se reajustó para dar cabida a los datos avanzados que se pretendían extraer de Zabbix, cumpliendo con la tarea 2c. Además, se instaló la aplicación en las HoloLens y se realizaron diversas pruebas, cumpliendo así con la tarea 2d.

*Asignación de horas por recurso:*

- Programador: 60h
- Tester: 20h

### Hito fase 2

Una vez finalizado el sprint 5, se obtuvo un segundo prototipo que fue aceptado por los directores, siendo presente en este el establecimiento de una conexión directa con el servidor Zabbix y la obtención de diversos datos de cada servidor localizado en el XML, aunque sin una interfaz de usuario detallada.

### Sprint 6. Diseño de interfaces de usuario

Tras obtener todos los datos necesarios desde Zabbix, se trabajó en el diseño de las interfaces gráficas del usuario (tareas 3a y 3b) que pudiesen adaptarse a la representación de dichos datos, y las interfaces auxiliares (tarea 3c) que definiesen el flujo del usuario a través de la plataforma.

*Asignación de horas por recurso:*

- Diseñador Software: 20h
- Diseñador Gráfico: 60h

### Hito fase 3

Una vez finalizado el sprint 6 no se obtuvo un prototipo nuevo, sino que se realizó una reunión con los directores en la que aceptaron los diseños preparados para su implementación posterior.

### Sprint 7. Implementación de interfaces con MRTK

Una vez obtenido el diseño, se implementaron las diversas interfaces (tareas 4a, 4b y 4c), empleando para ello la librería auxiliar MRTK (sección 3.2.2). La complejidad de las interfaces y el proceso de aprendizaje de la librería provocaron que el sprint se retrase semana y media. Posteriormente se instaló y comprobó en las HoloLens (tarea 4d).

*Asignación de horas por recurso:*

- Programador: 110h
- Tester: 30h

#### Hito fase 4

Una vez finalizado el sprint 7 se obtuvo un prototipo nuevo, el cual si bien los directores aceptaron como correcto, también sugirieron modificaciones que deberían ser llevadas a cabo en sprints posteriores.

#### Sprint 8. Separación del proyecto en escenas y ajustes a las interfaces

Se implementaron los cambios sugeridos por los directores en el anterior hito (4), tras lo cual se realizó una separación del proyecto (tareas 5a y 5b) en dos escenas de Unity diferentes, para poder concentrar cada una en una tarea en concreto (configuración inicial y muestra de gráficos y datos). Tras eso, se instaló la aplicación y se realizaron pruebas en las HoloLens (tarea 5c).

*Asignación de horas por recurso:*

- Diseñador Software: 20h
- Diseñador Gráfico: 10h
- Programador: 50h

#### Hito fase 5

Una vez finalizado el sprint 8 se obtuvo un prototipo nuevo, con la estructura final del proyecto ya definida, que contó con el visto bueno de los directores.

#### Sprint 9. Diseño de funciones extra

Durante la duración de este sprint se diseñaron las funciones auxiliares con comportamientos planteados en anteriores sprints con el objetivo de aportar al usuario de una cómoda experiencia al usar la aplicación (tareas 6a y 6b), las cuales se pueden comprobar en la sección 5.5.2.

*Asignación de horas por recurso:*

- Diseñador Software: 60h
- Programador: 20h

## Hito fase 6

Una vez finalizado el sprint 9 no se obtuvo un prototipo nuevo, sino que los directores revisaron y aceptaron los diseños planteados para las funciones auxiliares.

### Sprint 10. Implementación de funciones extra

En base a los diseños planteados en el anterior sprint, se implementaron las funcionalidades (tareas [7a](#) y [7b](#)) tal y como se puede comprobar en la sección [6.8.1](#). A continuación se instaló la aplicación en las HoloLens y se realizaron las pruebas necesarias (tarea [7c](#)).

*Asignación de horas por recurso:*

- Analista: 10h
- Programador: 50h
- Tester: 20h

## Hito fase 7

Una vez finalizado el sprint 10 se obtuvo un prototipo final, que cumplía tanto los objetivos iniciales como las funciones auxiliares del último sprint. Una vez obtenida la confirmación de los directores, este prototipo pasó a ser el entregable final del desarrollo de este trabajo de fin de máster.

### 4.3 Diagrama de Gantt

En este apartado se encuentra el diagrama de Gantt completo de la planificación del proyecto. Se muestran los hitos y los sprints, dentro de los cuales se sitúan las tareas del proyecto.

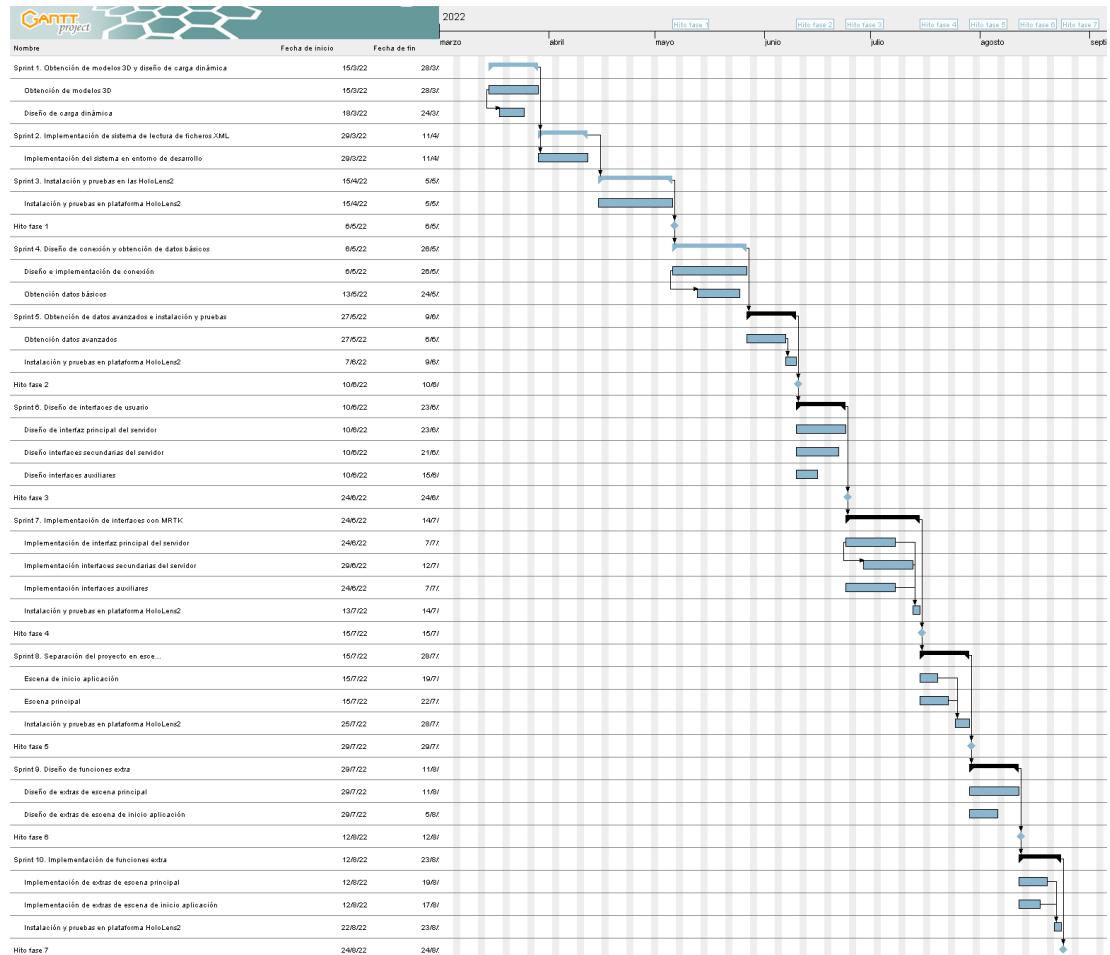


Figura 4.1: Diagrama completo de las tareas del proyecto. Creado con GanttProject

## 4.4 Estimación costes

Para realizar una estimación de costes aproximada del proyecto, se han tenido en cuenta las siguientes consideraciones:

- Aunque existan varios roles de proyecto, todos han sido realizados por un único recurso.
- Las horas computadas son las laborables, no las reales, pues el recurso ha trabajado en una empresa de software a jornada completa durante la duración de este proyecto.

### 4.4.1 Recursos materiales

Para todos los procesos del proyecto se emplearon los recursos descritos en el apartado 3.1, siendo su coste el siguiente:

- Equipo: Realizando una media de los precios actuales para dispositivos similares, y suponiendo una depreciación de 500€ debido al paso del tiempo (el PC data del año 2019), el coste de un producto con características similares ronda los **1800€**.
- HoloLens 2: A fecha de 2023, esta herramienta en la tienda oficial de Microsoft supone un coste de **3500€** en su versión de desarrollo [38]. Este dispositivo fue proporcionado por el CITIC para el desarrollo de este proyecto.

### 4.4.2 Recursos software

Debido a la utilización de software libre y/o licencias educativas, no existe un coste de recursos software asociados a este proyecto.

### 4.4.3 Recursos humanos

Los recursos humanos estimados son los siguientes:

- Analista: Es el encargado de analizar y extraer los requisitos del cliente o usuario final, y de establecer los objetivos principales del proyecto.
- Diseñador software: Es el encargado de realizar un diseño adaptado al problema concreto que la aplicación tiene que resolver.
- Programador: Es el encargado de implementar la aplicación, basándose siempre en el diseño previamente realizado.
- Diseñador gráfico: Es el encargado de diseñar las interfaces de usuario de tal forma que cumple los estándares básicos de accesibilidad.

- Tester: Es el encargado de diseñar y realizar todas las pruebas que permitan asegurar que el producto final cubre todos los requisitos establecidos.

La asignación de cada uno de estos perfiles se puede consultar en el detalle de cada sprint (sección 4.2.1).

#### 4.4.4 Costes finales

Para el cálculo de los salarios de los recursos humanos, se ha tomado como referencia el BOE-A-2023-6346 del XX Convenio colectivo nacional de empresas de ingeniería; oficinas de estudios técnicos; inspección, supervisión y control técnico y de calidad [39], ajustado al año 2022.

En la Tabla 4.1 se indican los costes estimados de los recursos humanos y en la Tabla 4.2 los costes de los recursos materiales:

Recurso	Coste/Hora	Hs Estimadas	Coste Total
Analista	40,00€/h	20h	800,00€
Programador	25,00€/h	560h	14.000,00€
Diseñador de Software	30,00€/h	180h	5.400,00€
Diseñador Gráfico	20,00€/h	70h	1.400,00€
Tester	10,00€/h	110h	1.100,00€
<b>Total</b>			<b>22.700,00€</b>

Tabla 4.1: Estimación de costes de los recursos humanos

Recursos materiales	Coste Total
Ordenador Portátil Asus Rog Strix	1.800,00€
HoloLens 2 Developer Version	3.500,00€
<b>Total</b>	<b>5.300,00€</b>

Tabla 4.2: Estimación de costes de los recursos materiales

Las horas aplicadas por cada recurso es una aproximación relativa al reparto de tareas en los sprints (sección 4.2.1) que se puede consultar en el diagrama de Gantt (Sección 4.3).

El coste total estimado del proyecto, tras sumar los costes materiales y los costes humanos, asciende a **28.000,00€**.

## Capítulo 5

# Diseño funcional

---

EN este capítulo se expone el diseño final de la aplicación, específicamente desde el punto de vista funcional.

### 5.1 Introducción

Como resultado de varias iteraciones, el diseño final se compone de diversas estructuras que trabajan de forma conjunta o paralela. Para mayor claridad, se ha dividido en tres grandes bloques: la pantalla de inicio de la aplicación; la representación gráfica del entorno; y la representación de la información de los servidores virtualizados obtenida a partir de Zabbix.

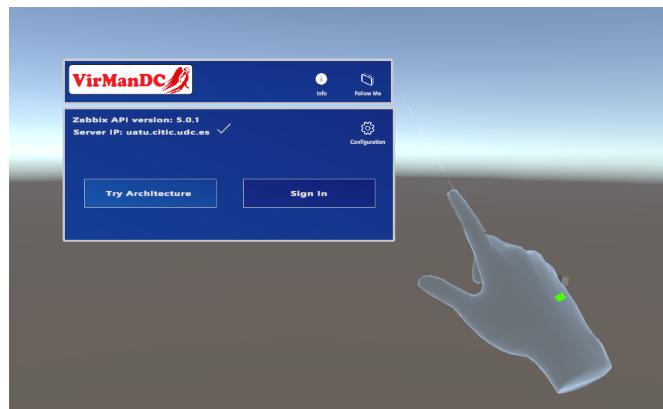


Figura 5.1: Captura de la pantalla de inicio

## 5.2 Pantalla de inicio

Al iniciar la aplicación se accede directamente a la pantalla de inicio (figura 5.1). Esta consta de una interfaz principal (sección 5.2.1), a partir de la cual se puede acceder tanto a la interfaz de opciones (sección 5.2.2) como a la interfaz de información (sección 5.2.3) empleando los botones habilitados para ello. Adicionalmente, se incluye una interfaz de error que sólo se activa en caso de que sucediera algún problema a la hora de intentar conectarse al servidor.

### 5.2.1 Interfaz inicial

Esta interfaz (figura 5.2) es la única que se encuentra activa desde el inicio de la aplicación. Consta de una barra de título, donde se encuentra el botón para acceder a la interfaz de información, y una sección mayor en la que se tiene acceso al botón para activar la interfaz de opciones, y los dos botones para ir a la siguiente pantalla del programa. Estos botones son los siguientes: el de *Check Architecture* permite acceder a la representación gráfica pero sin realizar conexión con Zabbix, por lo que parte de las funciones de la aplicación estarán limitadas; y el de *Log in* intenta establecer una conexión con Zabbix previamente a avanzar de pantalla.



Figura 5.2: Captura de la interfaz inicial

En esta interfaz también se encuentran elementos de información al usuario, entre los cuales está la URL del servidor al que se intenta conectar la aplicación, la versión de *API* de Zabbix que emplea dicho servidor o el indicador que muestra si la conexión se ha podido establecer correctamente.

### 5.2.2 Interfaz de opciones

Desde esta interfaz (figura 5.3) se pueden realizar diversos ajustes a la configuración de la aplicación, como modificar los parámetros de conexión a Zabbix, incluyendo IP y URL del servidor o las credenciales de la misma (usuario y contraseña). Estas últimas no se podrán leer directamente por el usuario y serán almacenadas tras ser cifradas.

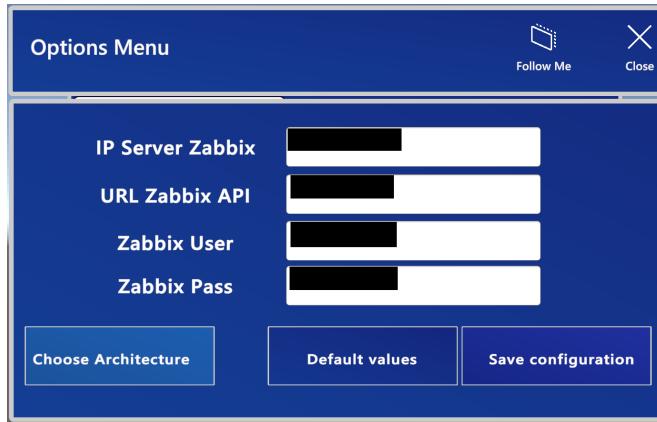


Figura 5.3: Captura de la interfaz de opciones

Los parámetros podrán ser guardados empleando el botón de *Save configuration*, el cual no sólo fijará los parámetros para esa sesión de la aplicación, sino que se almacenarán en un fichero dentro de las HoloLens. Para la comodidad del usuario, se pueden establecer unos valores predefinidos (a su vez modificables editando un fichero de configuración, explicado en detalle en el anexo A.4) a partir del botón *Default values*.

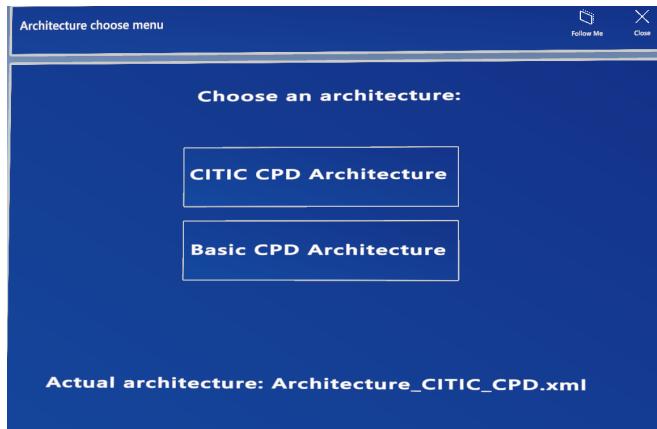


Figura 5.4: Captura de la subinterfaz de elección de arquitectura

Además del ajuste de los parámetros de la conexión, desde esta interfaz podemos acceder a la sub-interfaz (figura 5.4) de elección de arquitectura a través del botón *Choose Architecture*. En esta sub-interfaz aparecerán las diferentes arquitecturas que el programa tiene almacenadas en su carpeta de configuración (anexo A.3), pudiendo el usuario escoger cual será la que cargue posteriormente el programa.

### 5.2.3 Interfaz de información



Figura 5.5: Captura de la interfaz de información

Esta interfaz se activa desde el botón de ayuda de la interfaz principal, y permite acceder a un sencillo manual de instrucciones para familiarizarse con la herramienta.

## 5.3 Representación gráfica del entorno

Una vez atravesado el filtro de la pantalla inicial, se muestra la representación visual del CPD. La principal característica de esta representación es que emplea modelos 3D para realizar una réplica del CPD objetivo sin ser necesaria ningún tipo de conexión a Internet (habiendo seleccionado previamente la opción de "Check Architecture").

### 5.3.1 Lectura XMLs

Mediante la lectura de ficheros de configuración, es posible cambiar esa réplica sin tener que modificar la aplicación para ello. Simplemente modificando de forma externa dichos ficheros podemos variar no solo el CPD objetivo, sino también cambiar enteramente de CPD sin que la aplicación sufra algún tipo de modificación, siempre que los modelos 3D empleados hayan sido cargados previamente en el sistema.

### 5.3.2 Modelos

Dentro de los modelos 3D que se emplean en la aplicación, los principales son los siguientes:



Figura 5.6: Captura del modelo 3D de un armario rack

### Armario de rack

Los diferentes servidores se encuentran siempre en un armario especial, que contiene las conexiones de alimentación y las facilidades necesarias para manejar con comodidad dichos servidores, como son el espacio suficiente para cables de red, numeración lógica de los espacios verticales para los servidores (con una medida estándar en la industria, llamada U) o "rejillas" para desplazar los servidores en horizontal.

En esta aplicación el armario funciona como elemento agrupador, conteniendo los servidores de los cuales se mostrará la información; y como elemento familiar, pues así el usuario puede reconocer con más facilidad el entorno al que es asiduo.

### Servidor de rack



Figura 5.7: Captura del modelo 3D de un servidor de rack

Estos modelos se agrupan en el interior de los armarios de rack, representando fielmente su posición e interfaz externa. A su vez, funcionan como enlace para mostrar la información asociada a ese servidor. Pueden ser trasladados, rotados en eje horizontal y devueltos a su posición inicial, como se describe en la sección 5.3.4.

### 5.3.3 Elementos del entorno

Dentro de un CPD no sólo tenemos los componentes principales (armarios de rack y, en su interior, los servidores) sino que existen componentes adicionales que se encargan de proporcionar funcionalidad al CPD y de otorgar funcionalidades extra.

Algunos de estos elementos son por ejemplo los InRow, sistemas de refrigeración que se sitúan entre los armarios para proporcionar el enfriamiento necesario para el correcto funcionamiento de los servidores, o los SAI, sistemas que proporcionan electricidad a los servidores en caso de caída de la tensión de la red.

Estos componentes también son modelados en la aplicación, para otorgar de realismo al entorno virtual.

### 5.3.4 Funcionalidades de los modelos

Algunos modelos 3D permiten su interacción por parte del usuario, pudiendo modificar su rotación, escala y posición en el espacio virtual, empleando para ello componentes específicos de MRTK [40].

El primer modelo que permite estas operaciones dimensionales es el conjunto de los modelos 3D de armarios de rack. Para poder modificar este conjunto, se interactúa con la plataforma sobre los que se sitúan. No se permite la interacción con los armarios en solitario, pues eso agregaría una complejidad al usuario sin aportar valor extra.

Por otro lado, cada uno de los servidores individuales dentro de los armarios de rack permiten estas operaciones espaciales, ya sean monitorizables o no. Adicionalmente, se incorpora un botón de reinicio de posición, que permite devolver el servidor a su posición original dentro del modelo de rack.

## 5.4 Representación de la información de Zabbix

Una vez la conexión con el servidor Zabbix esté establecida, la aplicación es capaz de mostrar información de los servidores que estén siendo monitorizados por Zabbix. Específicamente, en esta aplicación se muestran datos de interés para el administrador de sistemas (uso de memoria, CPU, disco, etc.) para cada uno de los servidores que aparecen representados en el modelo, empleando para ello una interfaz de usuario interactiva que se activa una vez el usuario interactúa con el modelo 3D del servidor. Ésta muestra la información necesaria

de forma esquematizada, priorizando la más importante pero permitiendo el acceso a la más detallada con unas pocas interacciones del usuario.

#### 5.4.1 Información desde el modelo gráfico

Previamente a la interacción por parte del usuario sobre el servidor, para activar la interfaz principal de visualización de datos, la aplicación muestra una información valiosa sobre el estado del servidor.



Figura 5.8: Captura de la aplicación donde se aprecian dos servidores con alertas en su frontal

Concretamente, como se aprecia en la figura 5.8, los servidores que hayan detectado algún tipo de alerta mediante Zabbix tendrán en el frontal una señal de color rojo intenso, para informar al usuario de que existe algún problema. Estas alertas son detectadas al inicio de la escena, tras la carga de los gráficos, y se puede enviar una nueva petición mediante el comando auxiliar correspondiente, tal y como se explica en la sección 5.5.2.

Adicionalmente, mediante el comando auxiliar explicado en la sección 5.5.2, se pueden mostrar los servidores que están siendo monitorizados y, por lo tanto, pueden mostrar la información obtenida a través de Zabbix. En caso contrario, todavía se puede interactuar espacialmente con ellos.

#### 5.4.2 Interfaz principal

Al interactuar con un servidor monitorizado por Zabbix, se muestra la interfaz principal que se puede observar en la figura 5.9. Cuenta con cuatro segmentos diferenciados:

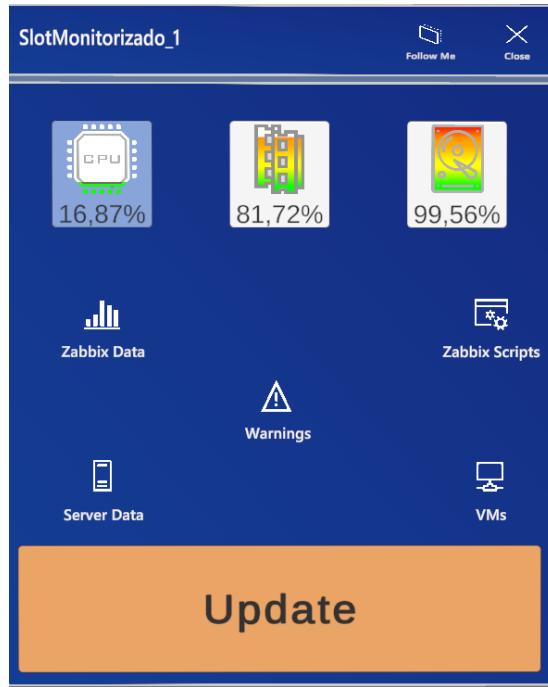


Figura 5.9: Captura de la interfaz principal del servidor

## Título

En la zona superior de la interfaz se encuentra el título, que consta del nombre del servidor siendo monitorizado y dos botones, uno para cambiar el comportamiento de seguimiento de la interfaz y otro que permite cerrarla.

## Indicadores

Inmediatamente después del título se encuentra la sección de indicadores, que tiene un número determinado por la configuración del modelo de interfaz. Estos indicadores muestran de forma inmediata la información más valiosa para el administrador de sistemas, como puede ser la carga de CPU o RAM del servidor, o el espacio de almacenamiento disponible del servidor. A su vez funcionan como botones de selección de datos a mostrar en la interfaz de datos de Zabbix (sección 5.4.5), ya que cada uno indica una categoría distinta de información.

## Botones de interfaces

Cada uno de ellos abre o cierra la interfaz correspondiente al nombre e icono que poseen.

### Botón de refresco

Este botón, en la zona inferior de la interfaz, permite enviar una nueva petición a la [API](#) de Zabbix para recuperar la información más reciente del servidor.

#### 5.4.3 Interfaz datos del servidor

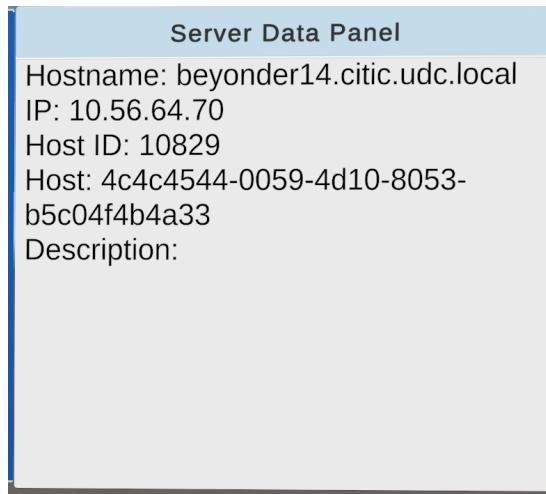


Figura 5.10: Captura de la interfaz de datos del servidor

Desde esta interfaz (figura 5.10) se comparte información básica del servidor, entre la que está incluida su IP, su Host ID (el identificador interno mediante el cual Zabbix clasifica a cada servidor), el nombre y la descripción definidos desde el servidor Zabbix.

#### 5.4.4 Interfaz de alertas

Zabbix es capaz de comprobar y emitir alertas si detecta que alguno de los parámetros monitorizados ha superado el límite preestablecido. En esta aplicación se representa de forma gráfica mediante un ícono en el frontal del servidor afectado (sección 5.4.1). Una vez interactuado con el servidor, las alertas detectadas se mostrarán en la interfaz correspondiente (figura 5.11). Si existe algún tipo de alerta, la interfaz de alertas se abrirá automáticamente al interactuar con el servidor, y el ícono del botón que abre la interfaz de alertas cambiará de color (figura 5.12).

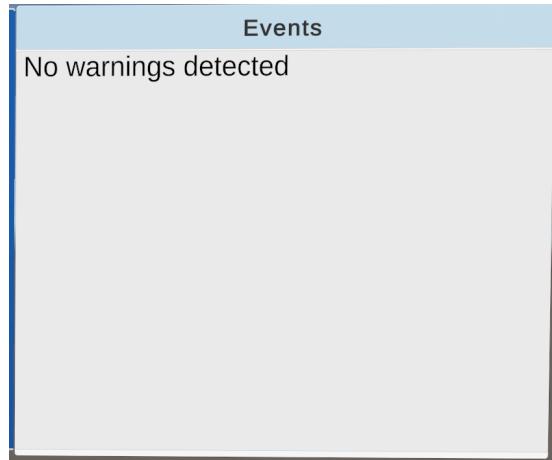


Figura 5.11: Captura de la interfaz de alertas del servidor

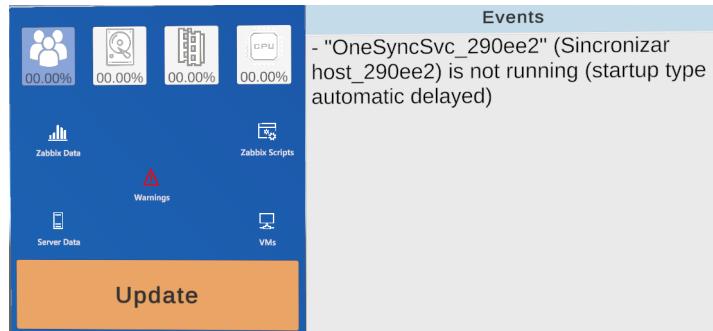


Figura 5.12: Captura de la interfaz de alertas del servidor con una alerta activa

#### 5.4.5 Interfaz de datos de Zabbix

Los indicadores de la interfaz principal (sección 5.4.2) muestran la información más importante al instante, pero si se necesita una información más detallada, al acceder a esta interfaz (figura 5.13) se muestran en formato de listado los datos ofrecidos por Zabbix. Para cambiar la categoría de los datos mostrados, se debe interactuar con los indicadores del menú principal.

#### 5.4.6 Interfaz de máquinas virtuales

Si el servidor monitorizado se trata de un hypervisor<sup>1</sup>, en esta interfaz (figura 5.14) aparecen las máquinas virtuales que gestiona en forma de listado. Al lado del nombre de cada una de las máquinas virtuales, aparece un indicador que muestra si la máquina está activa o no.

<sup>1</sup> Un hypervisor, conocido también como supervisor de máquina virtual (VMM), es un software que crea y ejecuta máquinas virtuales (VM) y que, además, aísla su sistema operativo y recursos de las máquinas virtuales y permite crearlas y gestionarlas.

Zabbix Data Panel		
Name	Description	Last Value
CPU usage	Aggregated CPU usage across all cores on the host in Hz. This is only available if the host is connected to a Zabbix agent.	31800000512
CPU frequency	The speed of the CPU cores. This is an average value if there are multiple speeds. The product of CPU usage and frequency is the total CPU power.	2444999936
CPU model	The CPU model.	AMD EPYC 7763 64 Core Processor
CPU cores	Number of physical CPU cores on the host. Physical CPU cores are the processors contained by a CPU package.	128
CPU threads	Number of physical CPU threads on the host.	256
CPU Total GHz		312959991808
CPU Usage percent of Total		332,96

Figura 5.13: Captura de la interfaz de datos de Zabbix del servidor

Al interactuar con uno de los elementos, se abrirá la interfaz principal de la máquina virtual (figura 5.15), que se diferencia de la interfaz de un servidor físico por el color rojo de fondo del título. La propia interfaz permite acceso a las demás interfaces con la información de la máquina virtual.

#### 5.4.7 Interfaz de ejecución de comandos

A través de la interfaz de comandos (figura 5.16) se tiene acceso a los comandos definidos en Zabbix y configurados en la aplicación. Consta de dos zonas diferenciadas, una con la lista de comandos accesibles y otra para la salida de dichos comandos.

### 5.5 User eXperience (UX)

En esta sección se pretende navegar por el flujo del usuario a lo largo de la aplicación y describir las funcionalidades auxiliares que no están incluidas para la representación de información o a la carga de gráficos.

#### 5.5.1 Menús auxiliares secundarios

Para ofrecer una experiencia agradable al usuario y coordinar correctamente la representación gráfica y de información de Zabbix, existen una serie de componentes que realizan determinados comportamientos destinados a mejorar la experiencia de usuario. La funcionalidad de estos componentes se agrupan bajo dos tipos de menús auxiliares:

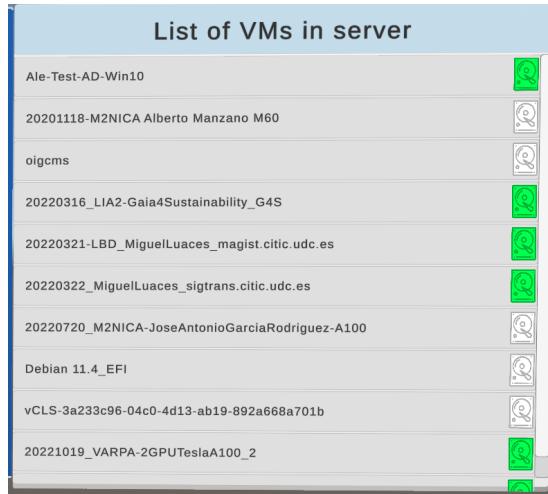


Figura 5.14: Captura de la interfaz de la lista de VMs del servidor

### Menú auxiliar de mano

Este menú aparece cuando se gira una de las manos mientras esté activa la representación gráfica (es decir, sólo tras el proceso de carga de los modelos 3D). Al dirigir la palma de la mano hacia la pantalla de las HoloLens, aparecerá un único botón, como se puede comprobar en la figura 5.17. El propósito de este botón es reiniciar la posición del menú auxiliar principal (sección 5.5.1), y colocar delante de la posición de los ojos del usuario. Este proceso es muy útil si se han movido componentes gráficos de tal forma que tapen la posición del menú. Al estar anclado a la mano del usuario, su acceso es cómodo.

### Menú auxiliar principal

Esta interfaz (figura 5.18) aparece junto con la representación gráfica, al cargar la arquitectura con los modelos 3D, y ofrece diferentes comportamientos auxiliares al usuario. Se puede reiniciar su posición tal y como se describe en la sección 5.5.1.

## 5.5.2 Funciones auxiliares

Las funcionalidades que encontramos disponibles en la interfaz auxiliar anteriormente mencionada son las siguientes:

### Info

Esta funcionalidad permite abrir una interfaz adicional, que como se aprecia en la figura 5.19 se utiliza para comparar una foto real del CPD con lo mostrado en la aplicación tras cargar los gráficos.

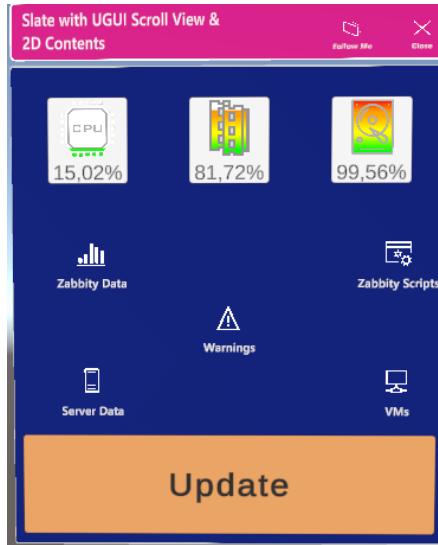


Figura 5.15: Captura de la interfaz principal de una máquina virtual

### Change Scale

Esta funcionalidad permite modificar el escalado de la representación del CPD, teniendo dos opciones: un escalado de mayor tamaño, el inicial, que permite apreciar los detalles de los servidores; y un segundo escalado menor para poder moverse con mayor comodidad por el modelo. Es un atajo para la funcionalidad descrita en la sección 5.3.4.

### Show Labels

Esta funcionalidad desactiva las etiquetas que muestran los nombres de los armarios rack. Se puede reactivar al volver a pulsar el botón.

### Update Warnings

Esta funcionalidad envía una petición de actualización de avisos (sección 5.4.4) de todos los servidores activos sin tener que interactuar con cada uno de ellos.

### Reload App

Esta funcionalidad permite reiniciar la aplicación sin tener que salir de la misma.

### Reset Position

Esta funcionalidad devuelve todos los servidores a su lugar de origen en el *rack* correspondiente. Es un atajo para la funcionalidad descrita en la sección 5.3.4.

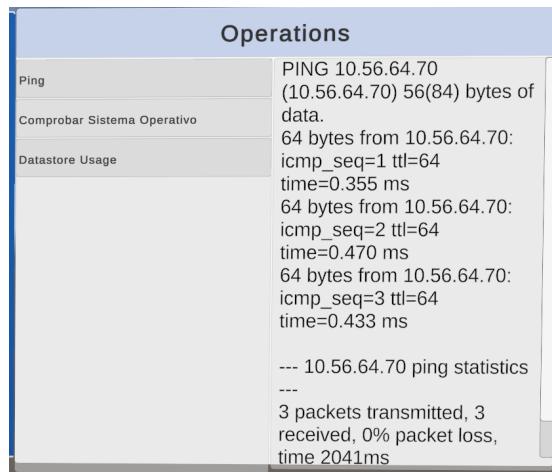


Figura 5.16: Captura de la interfaz de comandos del servidor

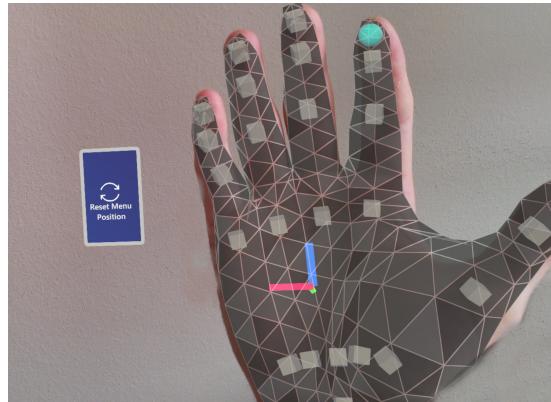


Figura 5.17: Captura del menú auxiliar de mano

### Show Guidelines

Esta funcionalidad activa la visión de unos elementos de unión entre el modelo 3D de un servidor físico y su interfaz de datos correspondiente. Es útil si se ha interactuado con varios servidores y no se ve claramente a qué interfaz corresponde cada uno. Se puede desactivar al volver a pulsar el botón.

### Show Inactives

Esta funcionalidad cambia el color a los frontales cuyos servidores no estén siendo monitorizados, para que el usuario pueda identificar qué servidores van a mostrar una interfaz al interactuar con ella. Se puede desactivar al volver a pulsar el botón.



Figura 5.18: Captura de la interfaz del menú auxiliar principal



Figura 5.19: Captura de la interfaz de información

## Rotate

Esta funcionalidad no aparece en un botón propiamente dicho, sino que consiste en un control deslizable, que permite rotar hasta un máximo de 360° grados el modelo completo del CPD. Es un atajo para la funcionalidad descrita en la sección 5.3.4.

### 5.5.3 Detección y muestra de errores

La detección de errores transcurre paralelamente a la ejecución del programa, y es útil para poder identificar y arreglar los problemas que surgen en el flujo del mismo.

Al ejecutar la aplicación en el entorno de desarrollo, estos errores se mostrarán en la interfaz de la consola de Unity, enviados tanto por el código de forma controlada como por el sistema al detectar un problema no contemplado. Pero para el usuario, los errores aparecerán en la interfaz de error (figura 5.20). La característica principal de esta interfaz es que estará presente únicamente si surge algún error en el sistema, ya sea de lectura de ficheros, de conexiones, de configuración incorrecta de los ficheros, etc.

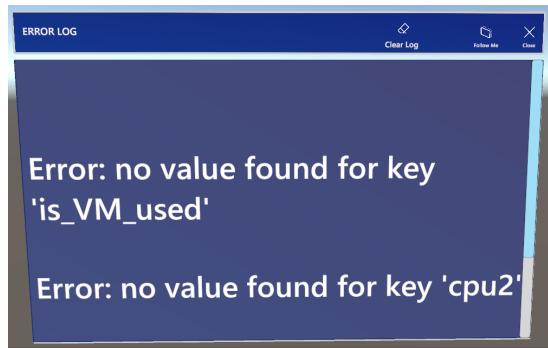


Figura 5.20: Captura de la interfaz de errores

El usuario podrá interactuar con ella de la misma manera que controla el resto de interfaces: es posible moverla en el espacio de la aplicación, cerrarla y permitir que siga al usuario en su movimiento en el plano real. De manera adicional, se puede limpiar la salida del interfaz para volver a repetir algún proceso que presente problemas y aislar el error a corregir.

#### 5.5.4 Flujo de UX

Para ilustrar el flujo del User eXperience, se muestra en la figura 5.21 un esquema que sintetiza todos los posibles pasos del usuario a través de las distintas interfaces.

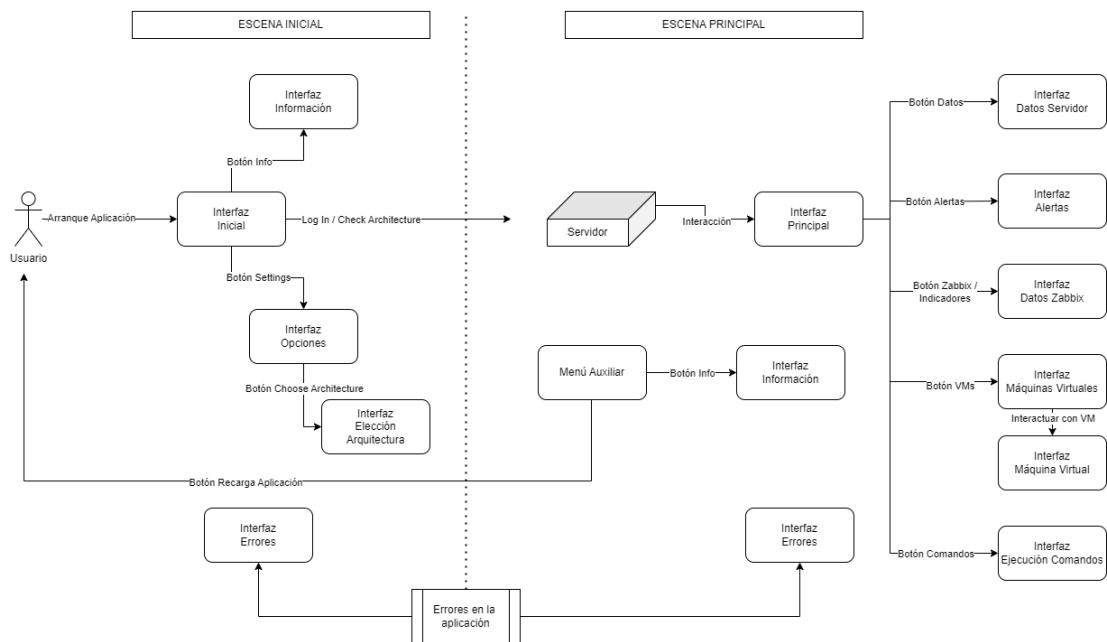


Figura 5.21: Diagrama de flujo del User eXperience



## Capítulo 6

# Diseño técnico

---

**E**n este capítulo se expone el diseño final de la aplicación, específicamente desde el punto de vista técnico.

### 6.1 Introducción

Una vez se ha descrito el funcionamiento de la aplicación y cuál es su diseño funcional final, se procede a describir los detalles en cuanto al diseño técnico del mismo.

### 6.2 Metodología de la implementación

Se considera de importancia dejar constancia de la metodología seguida a la hora de implementar el código de la aplicación, para poder seguir con claridad los pasos explicados en este capítulo. Se explica la organización de las carpetas del proyecto, la estructuración de la jerarquía de los objetos en Unity y los estándares y convenciones seguidos para el código.

#### 6.2.1 Organización del proyecto de Unity

Es importante conocer la organización de las carpetas dentro del proyecto Unity para ser capaz de introducir nuevos modelos 3D o frontales, además de explorar el código y las interfaces implementadas. Siguiendo las recomendaciones oficiales de Unity [41], se han estructurado las carpetas del proyecto de la siguiente manera:

Como se aprecia en la figura 6.1, la carpeta raíz de los recursos de la aplicación (llamada *Assets*) tiene varias carpetas, entre las cuales las de mayor importancia son:

- **Resources:** Carpeta especial de Unity [42] desde la que se cargan dinámicamente los recursos instanciados en tiempo de ejecución (sección 7.1.3), como son los modelos 3D o los frontales de los servidores.

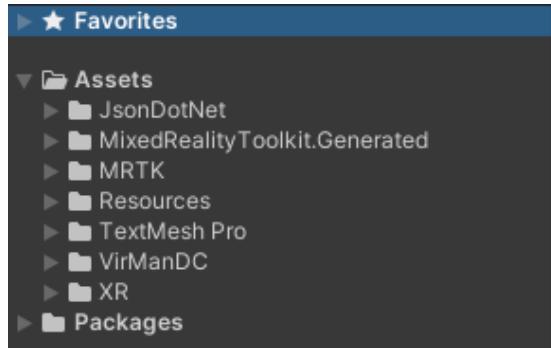


Figura 6.1: Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta Assets

- **VirManDC:** La propia carpeta del proyecto, organizada de acuerdo con los estándares de Unity.

El resto de carpetas son autogeneradas por las librerías empleadas por el proyecto, como Json.Net o MRTK.

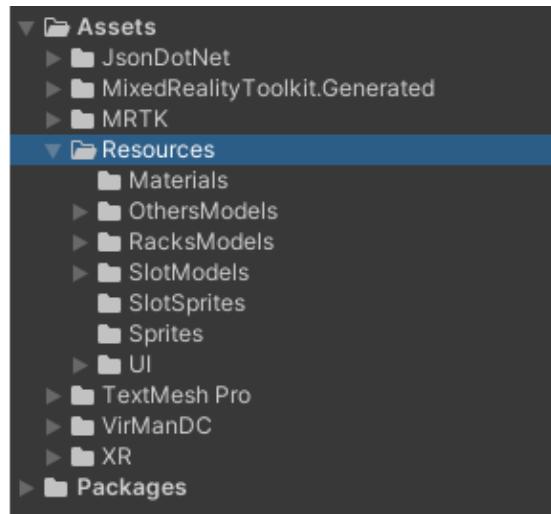


Figura 6.2: Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta Resources

El contenido de la carpeta **Resources** se aprecia en la figura 6.2:

- **Materials:** Los materiales empleados por los recursos de la carpeta principal.
- **OtherModels:** Los modelos 3D de los elementos del entorno (sección 5.3.3).
- **RacksModels:** Los modelos 3D de los racks (sección 5.3.2).
- **SlotModels:** Los modelos 3D de los servidores (sección 5.3.2).

- **SlotSprites:** Los gráficos de los frontales de servidores (sección 6.4.2).
- **Sprites:** Elementos gráficos de elementos de UI dinámicos (como los indicadores de las interfaces).
- **UI:** Elementos de interfaces de usuario.

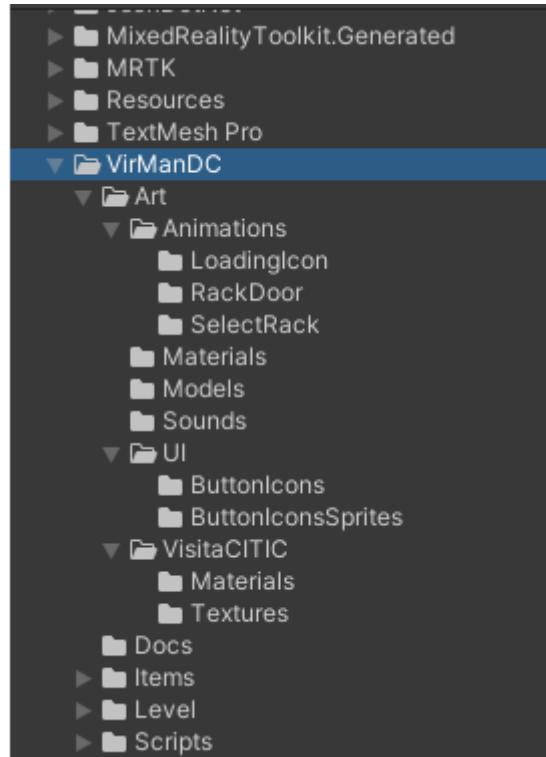


Figura 6.3: Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta VirManDC

El contenido de la carpeta **VirManDC** se aprecia en la figura 6.3:

- **Art:** Incluye todos los recursos gráficos que no se cargan de forma dinámica. Contiene las siguientes subcarpetas:
  - Animations: Incluye las diferentes animaciones de los modelos y las interfaces.
  - Materials: Los materiales de los elementos gráficos.
  - Models: Los recursos gráficos de los modelos 3D.
  - Sounds: Los recursos de sonido del proyecto.
  - UI: Los recursos gráficos de elementos de la interfaz, como los botones.

- VisitaCITIC: Incluye los recursos gráficos cedidos por la investigadora Aida Vidal Balea (sección 6.4.1).
- **Docs:** Incluye la documentación del proyecto, como la guía de Usuario.
- **Items:** Incluye recursos itemizables.
- **Level:** Incluye modelos 3D instanciados estáticamente y las escenas del proyecto.
- **Scripts:** Incluye el código del proyecto, con organización propia.

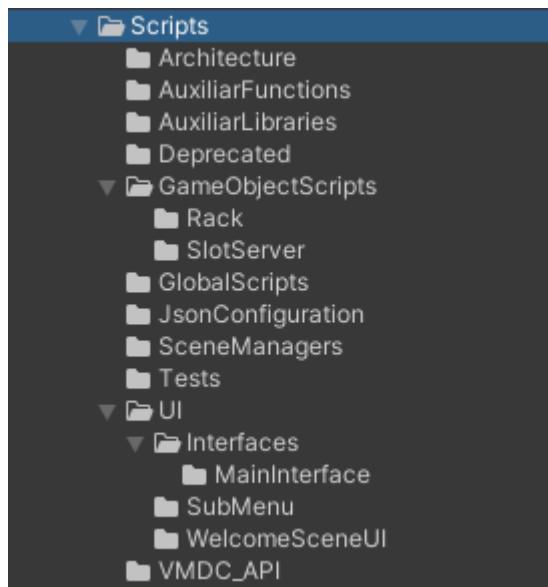


Figura 6.4: Captura de la interfaz de Unity en la que se aprecia el contenido de la carpeta Scripts

El contenido de la carpeta **Scripts** se aprecia en la figura 6.4:

- **Architecture:** Contiene los scripts encargados de la carga dinámica de los modelos 3D que conforman la arquitectura del CPD (sección 6.5).
- **AuxiliarFunctions:** Controladores asociados a las funciones auxiliares de la herramienta (sección 6.8.2).
- **AuxiliarLibraries:** Contiene los scripts de librerías empleadas en el proyecto importadas de manera directa, como la encargada del cifrado de las credenciales.
- **Deprecated:** Contiene los scripts descartados o en desuso, que se conservan como histórico si son de importancia suficiente.

- **GameObjectsScripts**: Contiene los scripts asociados al comportamiento de los objetos de la escena, como los modelos 3D, la base sobre la que se sitúan, etc.
- **GlobalScripts**: Contiene los controladores accesibles desde cualquier escena.
- **JsonConfiguration**: Contiene los scripts referidos a la carga de la configuración mediante JSONs.
- **SceneManagers**: Contiene los scripts que controlan el comportamiento de las escenas y las transiciones entre ellas.
- **Tests**: Contiene los controladores que permiten realizar acciones de depuración en el proyecto.
- **UI**: Contiene los scripts que controlan los comportamientos de las diferentes interfaces del proyecto, y de elementos relacionados como los paneles de inicio de sesión o cambios de configuración.
- **VMDC\_API**: Contiene los scripts relacionados con el establecimiento de la conexión con la API de Zabbix, conformando lo llamado API de VMDC (sección 6.6.3).

Aunque no forme parte de la carpeta del proyecto, es importante la carpeta *Configuration-Files*, encargada de contener los archivos XML y Json necesarios para la correcta ejecución de la aplicación (para más información, sección 6.5 y A.2). Esta carpeta deberá ir en la ruta de *Persistent Data Folder*<sup>1</sup>, que varía dependiendo del dispositivo en el que se ejecuta la aplicación [43]. En el caso de las HoloLens 2, el proceso detallado se explica en el anexo A.2.

### 6.2.2 Jerarquía de objetos de Unity

En Unity, los objetos se agrupan de forma jerárquica, tal y como se hace en la mayoría de herramientas de modelado 3D. Esto otorga facilidad de comprensión a la hora de interpretar la estructura de un objeto complejo; sin embargo, al tener diferentes componentes de comportamiento (*scripts*), la dificultad del entendimiento del funcionamiento de los objetos puede aumentar.

Por ello, en este proyecto se ha trabajado con la siguiente convención, que asegura una facilidad de entendimiento del comportamiento de los objetos: En todo objeto complejo, los componentes que otorgan un comportamiento específico estarán agrupados bajo un único objeto padre llamado *Controllers*. A su vez, cada componente estará unido a un único objeto hijo, que tendrá el mismo nombre que el componente. La figura 6.5 muestra un ejemplo de esta organización jerárquica.

---

<sup>1</sup> La Carpeta de Datos Persistentes es el directorio en el que se almacena para cada aplicación los archivos que se quieren conservar al acabar la ejecución de la aplicación. Varía según el dispositivo en el que se ejecute.

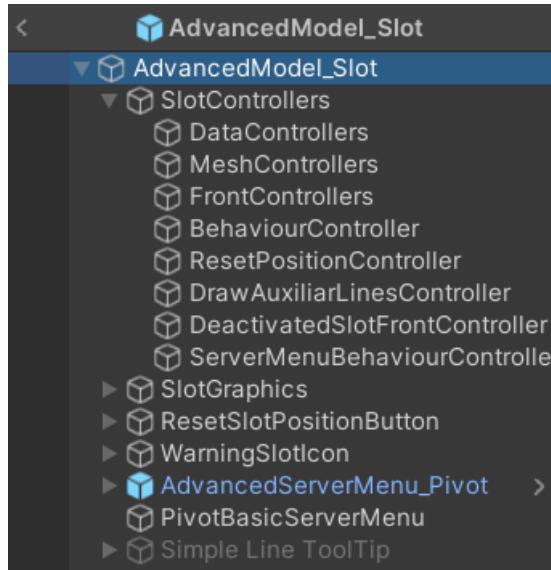


Figura 6.5: Captura de la interfaz de Unity en la que se aprecia la organización jerárquica del GameObject AdvancedModel\_slot

De ser necesario, un objeto complejo puede tener a su vez como hijos de jerarquía a otros objetos complejos, con comportamientos asociados, para los que se seguirá la misma convención. Esto sucede, por ejemplo, en las diferentes interfaces contenidas dentro de la interfaz principal de un servidor.

Es posible que existan excepciones, como por ejemplo el botón de reinicio de posición del servidor, debido al funcionamiento intrínseco de los componentes del MRTK.

### 6.2.3 Estándares de codificación y convenciones de nomenclatura del código

La codificación de los comportamientos de los objetos del proyecto se ha realizado completamente en C#, usando como lenguajes de apoyo XML y JSON, siguiendo las convenciones oficiales para este lenguaje de programación. Algunas de dichas convenciones son las de Microsoft .Net [44] y las internas de Google para C# [45]. Algunos ejemplos ampliamente utilizados a lo largo de todo el proyecto son los siguientes:

- Utilización de tipos específicos de excepciones para proporcionar mensajes de error significativos.
- Uso de declaraciones try-catch y using en el manejo de excepciones.
- Alineación del código de manera consistente para mejorar la legibilidad.
- Utilización de comentarios de una sola línea (//) para explicaciones breves.

- Nombres de clases, métodos, enumeraciones, campos públicos, propiedades públicas y espacios de nombres: PascalCase.
- Nombres de variables locales y parámetros: camelCase.
- Nombres de archivos y directorios son PascalCase, por ejemplo, MiArchivo.cs.

También se han empleado diversos **patrones de diseño** [46]. Entre ellos encontramos el *Fachada* (la clase *SlotDataFromAPI\_Manager*), el *Observer* (la clase *ArchitectureObjectsManager* mediante el uso de *actions*), el *Singleton* (los scripts de acceso global) o el *State* (el controlador *PanelsLocalManager*). Se utilizan en aquellas clases donde son necesarios, respetando siempre el sistema de componentes sobre el que está creado Unity. Es decir, el patrón *Componente* es intrínseco al proyecto y algunas posibilidades de diseño se ven mermadas, como las basadas en herencias (Unity obliga a heredar de la clase *MonoBehavior* [47] en la mayoría de scripts).

En esta memoria se hace referencia a las clases encargadas de ejecutar cada comportamiento definido, para consultar el código específico de cada clase consulte el repositorio oficial del proyecto [48]. Las clases se podrán encontrar en la carpeta **Scripts** del proyecto, siguiendo la organización planteada en el apartado 6.2.1. Dichas clases se mostrarán con el formato *NombreDeClase*, para facilitar su búsqueda.

### 6.3 Pantalla de inicio

La pantalla de inicio cuenta con varios controladores asociados:

- *StartController\_WelcomeScene*: Este script contiene referencias a las interfaces de la escena e inicializa la misma, realizando comprobaciones de errores de configuración.
- *LogInUIElementsController*: Se encarga de controlar los elementos de la pantalla de inicio de sesión, como el texto que indica la IP del servidor de Zabbix al que se intenta conectar el usuario.
- *Tester*: Determinados procesos de depuración de la escena.
- *LoadingSceneController*: Comportamientos relacionados con la carga de la escena principal.
- *ErrorManagerGUI*: Indica a qué interfaz se referencian los errores detectados.
- *ZabbixPetitions*: Clase encargada de realizar las peticiones a la API de Zabbix.
- *SlotDataFromAPI\_Manager*: Clase que sirve como puente entre las peticiones a la API de Zabbix y los elementos de la escena.

- *ChangeAppMode*: Permite cambiar el modo de visualización del programa (sólo arquitectura / mostrando datos).
- *IndicatorPanelManager*: Manager que realiza cargas de configuraciones a utilizar por los indicadores de los modelos.
- *ChooseArchitectureController*: Muestra al usuario qué modo de visualización está activo.
- *SettingsMenuController*: Esta clase se encarga del comportamiento de los elementos de UI en la interfaz de configuración.

### 6.3.1 Lectura y escritura del fichero de configuración

Desde la clase *StartController\_WelcomeScene* se realizan comprobaciones de ficheros de configuración (empleando la clase *ExtraClassesAndMethods\_VirManDC*).

Si se atraviesa ese filtro, se procede a realizar la carga del fichero de configuración desde la clase *VMDC\_AuxiliarConfiguration*. Esta última controla la generación de ficheros **JSON**, su acceso y edición, y el uso del fichero de valores por defecto.

### 6.3.2 Conexión con Zabbix

Si el proceso de lectura del fichero de configuración (sección 6.3.1) finaliza con éxito, el programa tratará de establecer una conexión con la API de Zabbix configurada según los parámetros del fichero, manteniendo la integridad de las credenciales mediante el uso de cifrado (sección 6.3.3).

En el caso de que la conexión se haya producido satisfactoriamente, se guarda la clave de autorización proporcionada por la API de Zabbix, para su posterior uso en la petición de datos específicos.

Si surge algún problema, el usuario es informado, y el programa inhabilita el acceso al entorno que muestra información de Zabbix. Sin embargo, el usuario mantiene la posibilidad de comprobar la arquitectura definida en los ficheros XML, pues para la representación gráfica no es necesario tener establecida una conexión.

### 6.3.3 Cifrado de credenciales

La implementación de un sistema de cifrado de las credenciales otorga una capa de seguridad extra a la aplicación, previniendo así ataques malintencionados que puedan intentar establecer una conexión ilícita con el servidor de Zabbix.

Si bien es un ámbito de gran extensión que podría aplicarse en un desarrollo completo aparte, se ha optado por utilizar un modelo básico de cifrado que, aunque no sea a prueba de los últimos mecanismos de descifrado de contraseñas, sí otorga una protección básica ante una posible brecha de los datos (por ejemplo, obtención del fichero de configuración para fines ilícitos).

Se ha utilizado una implementación básica de un algoritmo de cifrado basado en el [Estándar Avanzado de Encriptación \(AES\)](#), concretamente mediante Rijndael [49], cuya fuente original se encuentra en [StackOverflow](#) [50].

Se aplica una codificación en base64 que incluye bytes Salt<sup>2</sup> y IV<sup>3</sup> generados al azar, otorgando así a la codificación de un sencillo pero eficaz sistema de cifrado. Desde la llamada a la función, bastará con otorgar el texto a cifrar y la frase de cifrado, que estará embebida en el código del programa para evitar su robo de manera externa (la compilación del programa no permite la lectura del código fuente).

En la figura 6.6 se muestra el código principal empleado para el cifrado.

---

<sup>2</sup> En criptografía, salt es un conjunto de bits aleatorios que se usan como una de las entradas en una función derivadora de claves, que complican los ataques de diccionario que cifran cada una de las entradas del mismo: cada bit de salt duplica la cantidad de almacenamiento y computación requeridas

<sup>3</sup> En criptografía, el Vector de Inicialización (IV en inglés), se emplea para que un texto encriptado varias veces con la misma clave no siempre resulte en la misma cadena de texto

```

1 public static string Encrypt(string plainText, string passPhrase)
2 {
3     // Salt and IV is randomly generated each time, but is prepended
4     // to encrypted cipher text
5     // so that the same Salt and IV values can be used when decrypting.
6     var saltStringBytes = Generate256BitsOfRandomEntropy();
7     var ivStringBytes = Generate256BitsOfRandomEntropy();
8     var plainTextBytes = Encoding.UTF8.GetBytes(plainText);
9     using (var password = new Rfc2898DeriveBytes(passPhrase,
10         saltStringBytes, DerivationIterations))
11    {
12        var keyBytes = password.GetBytes(Keysize / 8);
13        using (var symmetricKey = new RijndaelManaged())
14        {
15            symmetricKey.BlockSize = 256;
16            symmetricKey.Mode = CipherMode.CBC;
17            symmetricKey.Padding = PaddingMode.PKCS7;
18            using (var encryptor = symmetricKey.CreateEncryptor(keyBytes,
19                ivStringBytes))
20            {
21                using (var memoryStream = new MemoryStream())
22                {
23                    using (var cryptoStream = new CryptoStream(memoryStream, encryptor,
24                        CryptoStreamMode.Write))
25                    {
26                        cryptoStream.Write(plainTextBytes, 0, plainTextBytes.Length);
27                        cryptoStream.FlushFinalBlock();
28                        // Create the final bytes as a concatenation of the random salt
29                        // bytes, the random iv bytes and the cipher bytes.
30                        var cipherTextBytes = saltStringBytes;
31                        cipherTextBytes = cipherTextBytes.Concat(ivStringBytes).ToArray();
32                        cipherTextBytes =
33                            cipherTextBytes.Concat(memoryStream.ToArray()).ToArray();
34                        memoryStream.Close();
35                        cryptoStream.Close();
36                        return Convert.ToString(cipherTextBytes);
37                    }
38                }
39            }
40        }
41    }
42 }
```

Figura 6.6: Código encargado de la encriptación de las credenciales

## 6.4 Importación de los gráficos

Para poder mostrar los componentes gráficos de los equipos del CPD, se deben obtener sus modelos y ser importados a Unity.

### 6.4.1 Modelos 3D

Dentro de los diversos modelos 3D que se emplean en la aplicación, los principales son los siguientes:

- **APC NetShelter SX 42U 600x1070**: Armario rack en el que se encuentran la mayoría de servidores.
- **Unidad de agua fría InRow RC, 200-240 V 50/60 Hz, IEC 309-16**: Unidad de refrigeración de armarios rack.

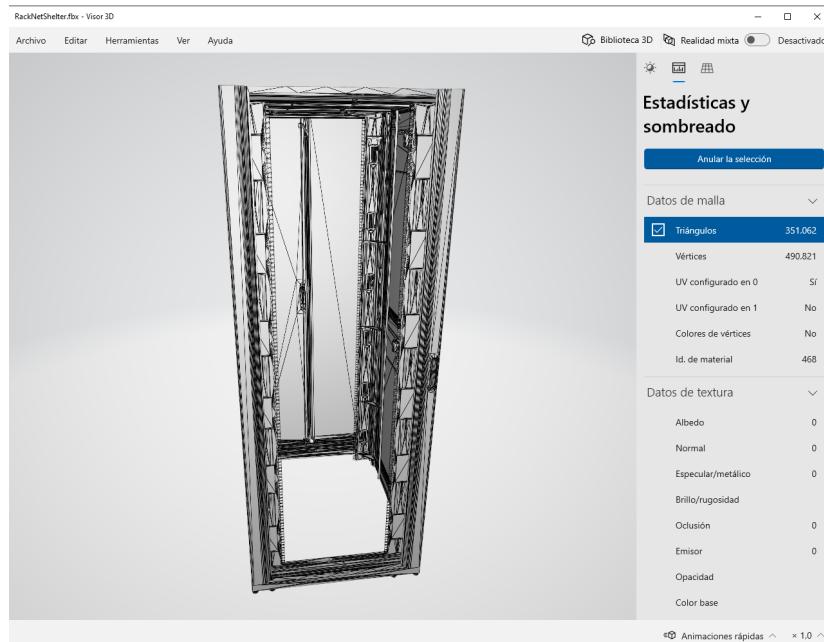


Figura 6.7: Modelo 3D detallado del armario de rack APC NetShelter SX 42U

La obtención de los modelos más detallados (figura 6.7), que se encuentran alojados en SketchFab, son de pago. Además, la versión de prueba muestra que son modelos 3D muy complejos, con un gran número de polígonos, adecuados para renderizados en alta calidad de imágenes estáticas, pero muy poco eficientes para su uso en una aplicación gráfica de renderizado continuo. De hecho, la optimización de los modelos 3D es una de las características básicas recomendadas en la documentación oficial de MRTK[51].

Tras comprobar que su uso ralentiza demasiado la aplicación, y la descomposición y simplificación del modelo conlleva un extenso proceso, se ha preferido crear un modelo propio que imite en los aspectos principales al modelo original.

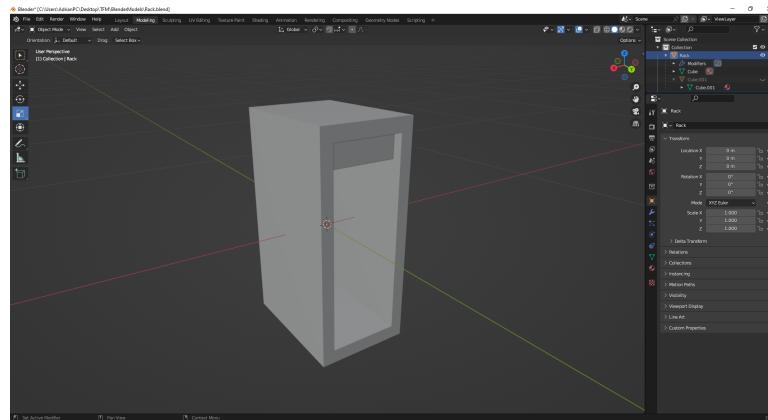


Figura 6.8: Modelo 3D básico del armario de rack

Para ello, se ha creado una réplica mediante el programa Blender, que ya incluye facilidades para exportar los modelos directamente a Unity. Si bien este modelo es de una complejidad mucho menor, y se ha utilizado durante varios sprints del proyecto, como se puede apreciar en la figura 6.8 es muy básico y visualmente no muy atractivo.

Gracias a la investigadora del GTEC Aida Vidal Balea, se ha podido reutilizar un modelo 3D intermedio de un proyecto del CITIC, y es este modelo el finalmente empleado en la aplicación, pues es lo suficientemente sencillo para no ralentizar la herramienta pero otorga detalles visuales al usuario como las puertas de los armarios o un diseño más estilizado (figura 6.9).



Figura 6.9: Modelo 3D intermedio del armario de rack

### 6.4.2 Frontales de los servidores

De los servidores, la parte que se muestra es la llamada frontal, que es la que se encuentra de cara al armario Rack, fácilmente observable desde el exterior del propio armario. El resto del servidor no consideramos que sea de importancia, pues los laterales y las partes superior e inferior están ocultas dentro del armario. La parte posterior, donde se suelen encontrar los cables de conexión y alimentación, no suele apreciarse una vez está el servidor activo.

Para ser lo más fieles posible, primeramente se han definido los modelos de servidores que se hayan en el CPD del CITIC. Tras preguntar al responsable, se ha obtenido la siguiente lista:

- Dell EMC PowerEdge R740
- Dell EMC Unity 450F
- Chasis Lenovo n1200
- Lenovo NeXtScale nx360 M5
- Lenovo System x3650 M5
- Switch Lenovo G7028

En Internet se encuentra la página VisioCafe [52], que se define como *"Un sitio web independiente y sin ánimo de lucro que reúne colecciones de Visio de la industria de tecnologías de la información"*. El formato Visio es un tipo de fichero vectorial usado por Microsoft para guardar esquemas y gráficos en alta resolución [53]. Puesto que no podemos importar directamente el archivo Visio a Unity, los extraeremos mediante la aplicación de Visio, y los exportaremos como **PNG**, un formato sencillo que es leído por casi todas las herramientas actuales, y que mantiene las propiedades del esquema original.

Sin embargo, la mayor parte de los recursos están preparados para ser abiertos desde Visio 2003, una aplicación de 20 años de antigüedad que se encuentra en desuso, por lo que se ha optado por importar los archivos desde la aplicación online de Drawio [54], y desde ahí realizar la exportación a PNG.

Una vez se obtienen los esquemas en PNG, se incluyen en la carpeta de **Resources** llamada *SlotSprites* (sección 6.2.1), para ser empleados en los frontales de cada servidor según su configuración.

### 6.4.3 Controladores de los modelos 3D

Cada modelo 3D principal tiene comportamientos asociados, que se detallan a continuación:

#### Rack

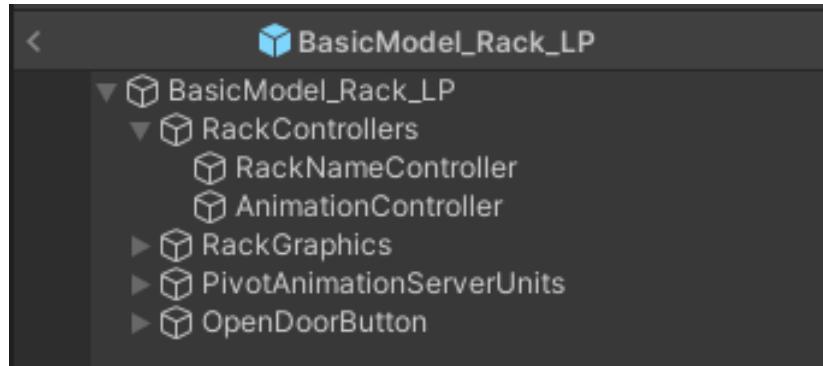


Figura 6.10: Captura del contenido del GameObject BasicModel\_Rack\_LP

Como se aprecia en la figura 6.10, el modelo 3D emplea los siguientes componentes:

- *RackNameController*: Encargado de otorgar a la etiqueta del modelo el nombre correcto.
- *AnimationController*: Proporciona las animaciones al modelo, como abrir la puerta.

Los botones poseen cierta lógica al ser elementos MRTK, tal y como se describe en el apartado 6.7.2.

#### Servidor

Como se aprecia en la figura 6.11, el modelo 3D emplea los siguientes componentes:

- *SlotData* y *SlotControl*: Funcionan como clases contenedores que almacenan la información del servidor, tanto la que proviene de la arquitectura como la de Zabbix.
- *FrontControllers*: Se encarga de controlar las interacciones con el frontal del servidor.
- *BehaviourController*: Se encarga de manejar los flujos de los datos entre las diferentes clases según el comportamiento del servidor.
- *ResetPositionController*: Proporciona la función auxiliar de reiniciar la posición del servidor (sección 5.5.2).
- *DrawAuxiliarLinesController*: Proporciona la función auxiliar de líneas de unión (sección 5.5.2).

- *DeactivatedSlotFrontController*: Proporciona la función auxiliar de informar si el servidor no está monitorizado (sección 5.5.2)
- *ServerMenuBehaviourController*: Controla el comportamiento de la interfaz principal del servidor

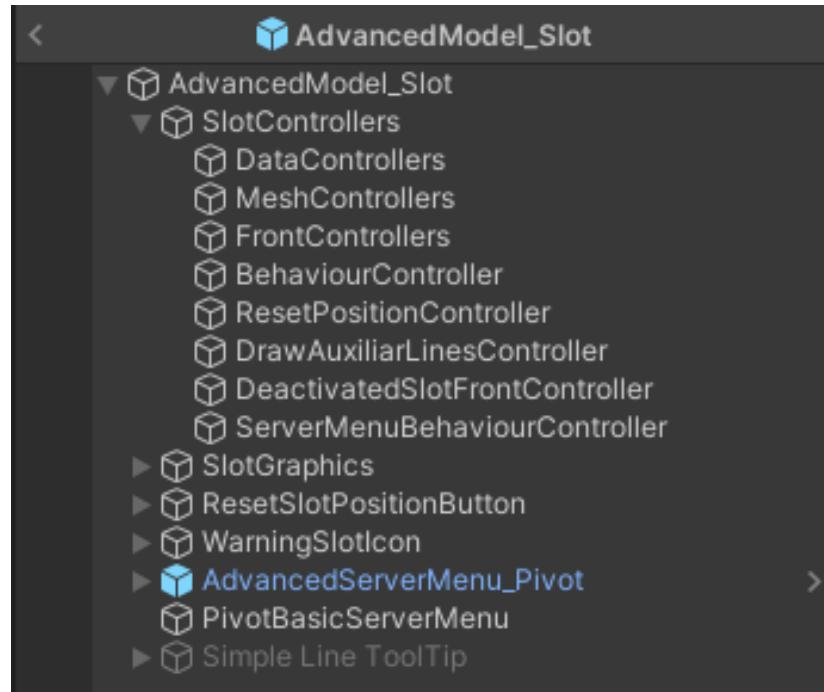


Figura 6.11: Captura del contenido del GameObject AdvancedModel\_Slot

Adicionalmente cuenta con el controlador *SlotComponentsReferences* en el objeto raíz, para poder ser accedido por elementos de la interfaz; y el objeto *WarningSlotIcon* tiene un comportamiento básico con un controlador asociado a ese objeto.

El botón de reinicio de posición posee cierta lógica al ser elemento MRTK, tal y como se describe en el apartado 6.7.2.

## 6.5 Carga de los gráficos mediante fichero XML

Una vez obtenidos los modelos 3D (sección 6.4), es necesario encontrar la forma de poder mostrarlos dinámicamente según la configuración deseada. Para ello, se empleará como base un fichero XML que contenga toda la información necesaria para la correcta muestra de los modelos.

Esta carga del fichero se realiza en memoria y se emplea al inicio de la escena principal, donde es llamado desde la clase *StartController\_MainScene*. Desde esta clase se llama a *ArchitectureGeneralManager*, que actúa como coordinador entre las clases *ArchitectureXmlManager* (encargada de la lectura directa de los ficheros y la traducción a DTOs<sup>4</sup>) y *ArchitectureObjectsManager* (encargada de la instanciación de los modelos a partir de los DTOs).

Para un detalle de cómo están diseñados los XML, consultar el anexo A.3.

## 6.6 Conexión de monitorización

La versión de Zabbix que monitoriza el CPD empleado como base para este proyecto tiene la versión **5.0.1**, la cual es estable ([LTS](#)) y relativamente moderna (la última versión estable disponible es la **6.4.0**) por lo que se tiene acceso a la mayor parte de las funciones que ofrece su API, pero no a todas. Consideramos que esto es conveniente, pues en un entorno real no siempre es posible tener disponible la última versión de un software y si esta aplicación es capaz de funcionar correctamente con las funciones más básicas, no se obliga al usuario final a tener su servicio de monitorización constantemente actualizado.

### 6.6.1 Conceptos de funcionamiento

Los elementos básicos en los que se descompone la conexión a la API de Zabbix son los siguientes:

- VPN: Utilizada para establecer conexión con el servidor de Zabbix, aunque en este proyecto finalmente no fue necesaria.
- VMDC API: Clases propias de la herramienta que se encargan de establecer la conexión directa con la API de Zabbix.
- Zabbix API: Interfaz de programación de aplicaciones ofrecida por Zabbix que permite consultar su información monitorizada previa autorización.

---

<sup>4</sup> Un DTO, o "Data Transfer Object", es una estructura de datos utilizada para representar información de manera simple y eficiente cuando se comunican sistemas o capas de una aplicación, transfiriendo datos entre diferentes partes de un programa sin exponer la estructura interna de los objetos subyacentes.

### 6.6.2 VPN

La conexión a la API de Zabbix se realiza hacia la entrada definida por el sistema, que suele ser una carpeta del servidor host del mánager de Zabbix. En este caso, ese servidor se encuentra en una red privada del CITIC, por lo que su acceso directo desde Internet no es posible.

En esta prueba de uso, y que suele ser equivalente a un caso de uso común en un entorno profesional, se empleó el software Pulse Secure (cambiado en 2023 a Ivanti Secure Access Client) [55] otorgado por la UDC, entidad de la que forma parte el CITIC, para establecer una VPN<sup>5</sup>. Adicionalmente se solicitó el acceso a la red privada y a la IP del servidor host de Zabbix, contactando con los servicios técnicos de la UDC.

En las primeras pruebas que se realizaron de esta aplicación, el software Pulse Secure estaba disponible también para las Hololens 2. Sin embargo, a mitad de proyecto las medidas de seguridad informática de la UDC cambiaron, y no se actualizó la aplicación en este entorno, por lo que la conexión por VPN quedó obsoleta.

Debido a esto, se tuvo que buscar una alternativa para la conexión: en lugar de conectar las gafas a una red wifi y luego emplear un software para establecer la conexión VPN, se utilizó un equipo adicional (el mismo que ya se utilizaba para desarrollar la aplicación y monitorizar las gafas) para servir de puente entre la VPN y las Hololens. Para ello, se probaron diferentes aplicaciones, pero finalmente se decidió emplear *Connectify Hotspot* por su facilidad de uso y gratuitad.

De esta manera, la VPN se establecía desde el computador con la aplicación Pulse Secure actualizada, y a su vez este equipo ofrecía un "hotspot" (punto de acceso wifi temporal) al que se conectaban las gafas, creando así un túnel entre las conexiones.

Sin embargo, esta solución no tardó en ser descartada pues el software no era muy fiable y fallaba con frecuencia, con lo que se contactó con el responsable del CITIC encargado de las conexiones y, finalmente, se abrió la URL del servidor Zabbix al dominio público. Por lo tanto, la conexión podía establecer independientemente de la red a la que estuviesen conectadas las gafas (con el único requisito de tener acceso a Internet), pero eso conllevaba la necesidad de una seguridad extra en el programa que se solventó con la inclusión del cifrado de las credenciales, tal y como se explica en la sección 6.3.3.

---

<sup>5</sup> VPN, o "Virtual Private Network", es una tecnología que crea una conexión segura y cifrada entre dispositivos o redes a través de Internet, ocultando la dirección IP real y encriptando los datos

### 6.6.3 VirManDC API

El conjunto de clases implementadas para establecer la conexión con Zabbix se agrupan en la carpeta *VMDC\_API*, siendo éstas las siguientes:

- *ResponseAPIClasses*: Agrupa las clases básicas que se emplean para serializar y deserializar los mensajes entre Zabbix y la herramienta, concretamente las respuestas.
- *SlotDataFromAPI\_Manager*: Clase que se encarga de ocultar la implementación de la API de Zabbix a las demás clases del proyecto. Es un patrón Fachada (Facade).
- *VMDC\_JsonDTOs*: Agrupa las clases básicas que se emplean para serializar y deserializar los mensajes entre Zabbix y la herramienta, concretamente las peticiones.
- *ZabbixPetitions*: Agrupa todas las peticiones que se realizan a la API de Zabbix, incluyendo peticiones de acceso, de alertas, de máquinas virtuales de un hipervisor, etc.

Todas ellas conforman el subsistema de VirManDC API, que se conecta y transmite la información de Zabbix a la aplicación.

### 6.6.4 Zabbix API

Si bien el formato y estructura de la *API* de Zabbix está detalladamente explicado en la documentación oficial [56], en este apartado se realiza una descripción básica.

El API de Zabbix permite modificar la configuración de Zabbix y obtener acceso a los datos históricos. Es esto último lo que aplicaremos en esta herramienta.

Esta API está basada en HTTP, y forma parte del frontend de la aplicación web de Zabbix. Emplea el protocolo JSON-RPC 2.0, por lo que se usará el formato *JSON* para las peticiones y las respuestas, y la API estará conformada por diferentes métodos.

En esta aplicación se emplea la clase de Unity *UnityWebRequest* para el envío de peticiones, tal y como se aprecia en la clase *ZabbixPetitions*.

Dentro de esta clase se encuentran todos los métodos de las peticiones a la API que emplea la herramienta, siendo los siguientes:

- *apiinfo.version*: Este método devuelve la versión de API que utiliza el servidor Zabbix, no es necesario estar autenticado.
- *user.login*: Método inicial que permite autenticar al usuario, devuelve una clave de sesión para poder realizar la mayoría de operaciones.
- *item.get*: Método básico para obtener información sobre un ítem de Zabbix. Se utiliza para obtener la información detallada del servidor (sección 5.4.5) y para conocer si las máquinas virtuales detectadas están activas o no (sección 5.4.6).

- *host.get*: Método que devuelve una lista de hosts. Se emplea para la obtención de datos básicos de los servidores (sección 5.4.3), y para la obtención de máquinas virtuales (sección 5.4.6).
- *script.execute*: Método que ejecuta y devuelve la salida de los scripts predefinidos en Zabbix (sección 5.4.7).
- *hostgroup.get*: Este método devuelve el grupo de Hosts al que pertenece determinado host. Se utiliza para la obtención de máquinas virtuales.
- *trigger.get*: Método que devuelve la lista de alertas asociadas a un determinado servidor (sección 5.4.4).

Para más información sobre el formato de las respuestas a las consultas, consultar la sección 6.6.5.

### 6.6.5 JSONs de respuesta

Como ya se comentó en la sección 6.6.4, los objetos devueltos por la API de Zabbix están en formato [JavaScript Object Notation \(JSON\)](#) [57], un tipo de fichero semiestructurado que permite la transmisión de información entre aplicaciones de forma cómoda y eficiente. Para ser utilizados correctamente, la aplicación debe ser capaz de serializar y deserializar (transformar objetos en cadenas de texto y viceversa) los JSON.

Este aspecto técnico se puede apreciar en la implementación de las conexiones con la API de Zabbix, pues las consultas son serializadas (transformadas en cadenas de texto a partir de los objetos empleados para la conexión, como son los diccionarios o las listas), tal y como se aprecia en la clase *ZabbixPetitions*.

A su vez, las respuestas de la API son deserializadas (transformadas en objetos empleados por la aplicación a partir de la cadena de texto recibida como respuesta).

Para ello, se emplea la librería *Newtonsoft.Json*, pues aunque el propio Unity posee un serializador interno[58], éste es muy básico y no tiene la capacidad de transformar correctamente los objetos necesarios para realizar consultas avanzadas. Existen otras librerías disponibles, integradas en la Store de Unity, pero son formatos de prueba o no son permitidos por el compilador de .Net de Visual Studio, y se detectan sus referencias como errores.

Por lo tanto, aunque el uso de la librería *Newtonsoft.Json* implica buscar una versión más antigua que la última disponible, es actualmente la única forma de obtener una correcta serialización de los objetos JSON.

## 6.7 Representación de la información de Zabbix

A partir de los objetos obtenidos como respuesta a la API de Zabbix (sección 6.6.4), podemos representar fielmente la información obtenida, repartido según su tipo en diferentes paneles (sección 5.4.2). Los paneles se encuentran dentro del Gameobject prefabricado (Prefab)<sup>6</sup> llamado *AdvancedServerMenu\_Pivot*, que se encuentra en la subcarpeta *UI* de *Resources*. Además de los controladores que manejan el comportamiento global de las interfaces (conexiones con la fuente de datos almacenados en otro Gameobject, apertura y cierre de los paneles, seguimiento del usuario para facilitar su manejo...), cada panel tendrá su propio controlador que recibirá la información del API, y la representará en su apartado visual correspondiente.

### 6.7.1 Información desde el modelo gráfico

Previamente a la interacción por parte del usuario sobre el servidor, para activar la interfaz principal de muestra de datos, la aplicación muestra una información valiosa sobre el estado del servidor, tal y como se explica en la sección 5.4.1.

Para ello se ejecutan las clases *WarningController*, que realiza una petición de refresco de los datos de alertas a la clase intermediaria *BehaviourSlotController*. Ésta, a través de la clase *SlotDataFromAPI\_Manager*, ejecuta la función *WarningsPetition* de la clase *ZabbixPetitions*.

Estas alertas son detectadas al inicio de la escena, tras la carga de los gráficos, y se puede enviar una nueva petición mediante el comando auxiliar correspondiente, tal y como se explica en la sección 6.8.2.

---

<sup>6</sup> Prefabricados, instancias de objetos o elementos predefinidos y reutilizables que pueden contener componentes, scripts y configuraciones específicas

### 6.7.2 Elementos empleados de MRTK

Para la confección de las interfaces, se han utilizado diversos objetos auxiliares de la librería MRTK. En concreto, finalmente se ha empleado la versión 2.8, pues es un desarrollo en progreso y cada pocos meses se publica una nueva versión.

#### Botones



Figura 6.12: Captura de HoloLens 2 con botones de MRTK

Los botones (figura 6.12) son los elementos básicos más empleados de las interfaces en este proyecto. Otorgan muchas ventajas, como son el aspecto visual mejorado, la mejora de la interacción del usuario con respecto a los botones por defecto de Unity, o la fácil configuración dentro de unos determinados parámetros.

Sin embargo, también posee algunas dificultades, como una elevada curva de aprendizaje inicial para su correcto uso (está conformado por diversos componentes propios de MRTK) o la poca personalización más allá de determinados parámetros (como los iconos de los botones). Por ejemplo, no existe una solución fácil para cambiar el color de los mismos, o representar gráficamente su estado desactivado, algo que sí permiten los botones más básicos de Unity.

Para un mayor detalle, consultar la documentación oficial [59].

### Menú cercano



Figura 6.13: Captura de HoloLens 2 con el menú cercano

El menú cercano (figura 6.13) proporciona una colección de botones con un comportamiento flotante sobre el usuario. En este proyecto se emplea en el menú auxiliar (6.8.1).

Para un mayor detalle, consultar la documentación oficial [60].

### Menú de mano

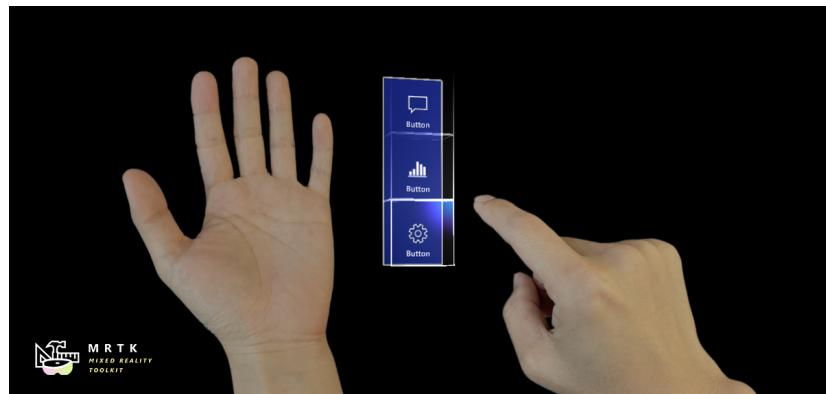


Figura 6.14: Captura de HoloLens 2 con el menú de mano

El menú de mano (figura 6.14) proporciona una colección de botones con un comportamiento flotante sobre el usuario. En este proyecto se emplea en el menú auxiliar de mano (6.8.1).

Para un mayor detalle, consultar la documentación oficial [61].

### Controles deslizantes

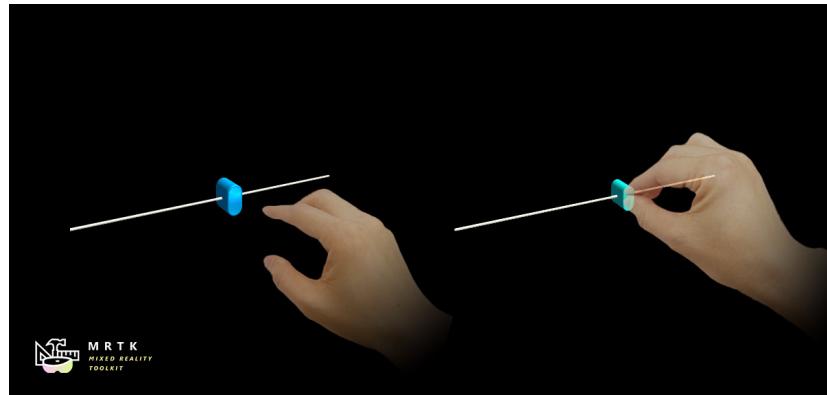


Figura 6.15: Captura de HoloLens 2 con control deslizante

El control deslizante (figura 6.15) proporciona un comportamiento de cambio de valor al mover un control deslizante. En este proyecto se emplea en el menú auxiliar (6.8.1).

Para un mayor detalle, consultar la documentación oficial [62].

### Información sobre herramientas

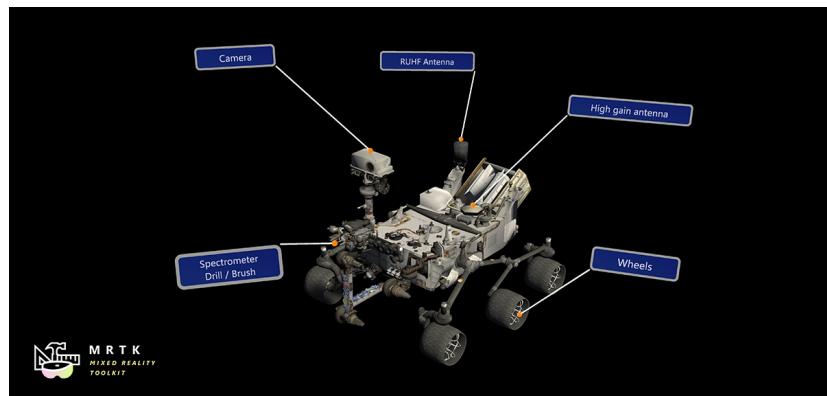


Figura 6.16: Captura de HoloLens 2 con el componente de información sobre herramientas

El componente de información sobre herramientas (figura 6.16) proporciona información al usuario al interactuar con un objeto del entorno. En este proyecto se emplea en la información de los racks (5.3.2).

Para un mayor detalle, consultar la documentación oficial [63].

## Interfaces MRTK y modificaciones

El conjunto de todas las interfaces de la primera escena, las interfaces de error e información de la escena principal y la interfaz principal de servidor están basadas en un prototipo de interfaz definido en uno de los proyectos de ejemplo de MRTK. Sobre este prototipo se han realizado modificaciones para adaptar su comportamiento al deseado para este proyecto; por ejemplo, el comportamiento de seguimiento de la interfaz al usuario ha sido cambiado por uno propio, llamado *CustomFollowMeController*, que emplea a su vez las clases *KeepRelativePositionToCamera* y *KeepPositionAndRotationController*.

Adicionalmente, se emplea el componente *System Keyboard* para la entrada del usuario y los controladores de manipulación para las operaciones espaciales de los modelos 3D (sección 5.3.4).

En definitiva, MRTK aporta diversos elementos que requieren de un aprendizaje previo de la librería, pero cuyo valor final suele ser superior a intentar replicar el comportamiento con elementos propios de Unity.

### 6.7.3 Interfaz principal

La interfaz principal está implementada a partir de un objeto de la librería MRTK, con los elementos y controladores propios del mismo.

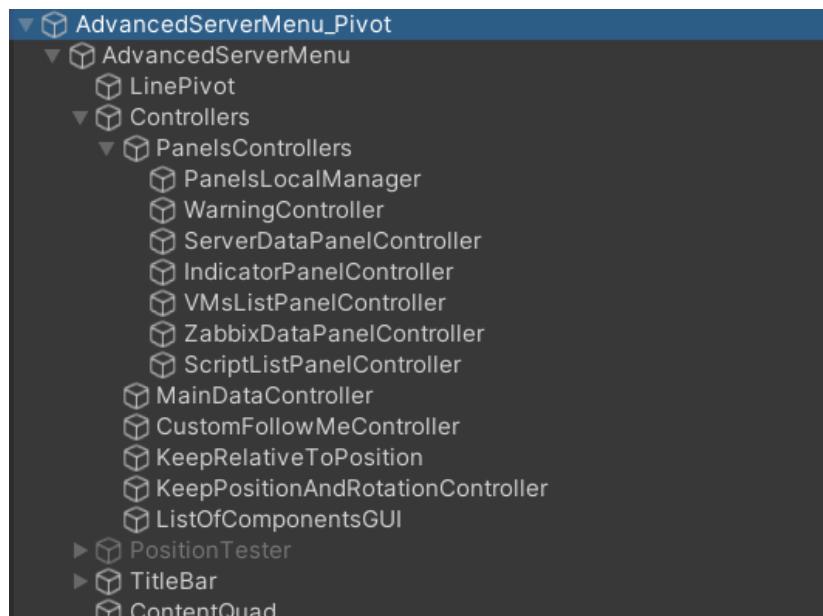


Figura 6.17: Captura de Unity en la que se aprecian los principales controladores de la interfaz principal

Por otro lado, cuenta con los componentes que se aprecian en la figura 6.17:

- *PanelsLocalManager*: Controla el comportamiento básico de las diferentes subinterfaces.
- *WarningController*: Empleado para realizar peticiones de actualización de alertas.
- *ServerDataPanelController*: Controla la subinterfaz de datos del servidor (sección 5.4.3).
- *IndicatorPanelController*: Controla el comportamiento de los indicadores de la interfaz.
- *VMsListPanelController*: Controla la subinterfaz de máquinas virtuales (sección 5.4.6).
- *ZabbixDataPanelController*: Controla la subinterfaz de datos de Zabbix (sección 5.4.5).
- *ScriptListPanelController*: Controla la subinterfaz de ejecución de comandos (sección 5.4.7).
- *MainDataController*: Empleado para realizar peticiones de actualización de información de Zabbix.
- *ListOfComponentsGUI*: Empleado para cambiar el nombre del servidor según esté definido.

Adicionalmente a los mencionados, existen componentes de control de interfaces cuyo funcionamiento se describe en la sección 6.7.2.

### Indicadores

Cada uno de los indicadores tiene un comportamiento definido en el propio objeto, pues es relativamente complejo y funciona como un entorno separado. Se puede consultar en detalle en el prefab *Indicator\_space\_v2*, incluido en la carpeta *UI* de *Resources*. Este prefab incluye el componente *IndicatorController*, encargado de la lógica específica del indicador.

### Botones de interfaces

El comportamiento de cada botón es trasladado al componente *PanelsLocalManager* de la interfaz principal. Cada botón cuenta con su propia lógica al ser un botón MRTK.

### Botón de refresco

Este botón emplea los componentes *MainDataController* y *WarningController* de la interfaz principal para ejecutar tareas de actualización de datos. Al ser necesario un cambio en el formato visual, se empleó un botón propio de Unity.

#### 6.7.4 Subinterfaces del servidor

La lógica de cada una de las subinterfaces se encuentran en los componentes de la interfaz principal (sección 6.7.3). Como detalle importante, estas subinterfaces se implementaron mediante elementos de Unity, no tiene ningún componente asociado de MRTK pues su excesiva complejidad no permite realizar ciertos ajustes necesarios para el correcto planteamiento de cada subinterfaz.

### 6.8 User eXperience (UX)

#### 6.8.1 Menús auxiliares secundarios

Para ofrecer una experiencia agradable al usuario y coordinar correctamente la representación gráfica y de información de Zabbix, existen una serie de componentes que realizan determinados comportamientos destinados a mejorar la experiencia de usuario. La implementación de estos componentes se agrupan bajo dos tipos de menús auxiliares:

##### Menú auxiliar de mano

Como ya se mencionó en la sección 6.7.2, este menú está basado en un componente MRTK, por lo que cuenta con lógica propia. Se han modificado ciertas características para adaptarlo al proyecto, como la eliminación de ciertos botones. El comportamiento asociado al botón se encuentra en el componente *KeepPositionAndRotationController* del menú auxiliar principal (sección 6.8.1).

## Menú auxiliar principal

Como ya se mencionó en la sección 6.7.2, este menú está basado en un componente MRTK, por lo que cuenta con lógica propia. Se han modificado ciertas características para adaptarlo al proyecto, como la inclusión de un control deslizante o un incremento de los botones. Se encuentra como hijo del objeto raíz *BaseObject* de la escena principal.

Los comportamientos asociados a los botones están definidos en los componentes que se encuentran en el objeto *Controllers* de la escena principal. Cada uno de estos componentes se asocia a una función auxiliar.

### 6.8.2 Funciones auxiliares

#### Info

Este comportamiento viene dado por el elemento *VisualsMenuController*, que se encuentra en el objeto *Controllers* de la escena principal.

#### Change Scale

Este comportamiento viene dado por el elemento *ScaleBaseController*, que se encuentra en el objeto *BaseObject* de la escena principal.

#### Show Labels

Este comportamiento viene dado por el elemento *RackManagers*, que se encuentra en el objeto *Controllers* de la escena principal.

#### Update Warnings

Este comportamiento viene dado por el elemento *AllWarningManager*, que se encuentra en el objeto *Controllers* de la escena principal.

#### Reload App

Este comportamiento viene dado por el elemento *SceneManager*, que se encuentra en el objeto *Controllers* de la escena principal.

#### Reset Position

Este comportamiento viene dado por el elemento *ResetAllSlotsPositionManager*, que se encuentra en el objeto *Controllers* de la escena principal.

### Show Guidelines

Este comportamiento viene dado por el elemento *GuideLinesManager*, que se encuentra en el objeto *Controllers* de la escena principal.

### Show Inactives

Este comportamiento viene dado por el elemento *ShowDeactivatedSlotsManager*, que se encuentra en el objeto *Controllers* de la escena principal.

### Rotate

Este comportamiento viene dado por el elemento *RotationBaseController*, que se encuentra en el objeto *BaseObject* de la escena principal.

#### 6.8.3 Detección y muestra de errores

Cada posible salida de error detectada durante la implementación de este proyecto, ya sea por fallos en la configuración por parte del usuario (conexiones, ficheros) o por la propia implementación de la herramienta, es controlada mediante la clase global *ErrorManager*. A partir de un sistema de eventos propio de C#, cada vez que se detecta un error esta clase envía el contenido del mensaje de error a la clase *ErrorManagerGUI*, que cuenta con las referencias necesarias para enviar este error directamente a una interfaz de usuario. Así, cualquier error detectado estará disponible para el usuario en la plataforma, sin tener que depender de ficheros de registros adicionales o depuración de la aplicación desde los entornos de desarrollo.

## Capítulo 7

# Desarrollo y pruebas

EN este capítulo se describen las principales características destacables del desarrollo del proyecto y las diferentes pruebas realizadas para comprobar la viabilidad del producto.

## 7.1 Desarrollo en PC y Hololens

El desarrollo principal de la aplicación se realizó desde el PC, con las herramientas expuestas en el capítulo 3. Puesto que la ejecución de la aplicación en las HoloLens conllevaba una compilación completa (consultar anexo A.1 para más detalles), la mayor parte del desarrollo y de las pruebas asociadas se realizaron desde el entorno Unity, pues posee un depurador muy potente y directo. El resultado de este desarrollo puede verse en la figura 7.1.



Figura 7.1: Captura de la aplicación VirManDC en la que se aprecian los modelos 3D y la interfaz principal de servidor

### 7.1.1 Entorno novedoso

Debido a la novedad de las HoloLens 2, su elevado precio y su escasa adopción por usuarios comunes, es difícil encontrar documentación y material estable o comprobado. Es destacable, por tanto, que a lo largo de este proyecto existieron diversos problemas a los que el equipo de desarrollo tuvo que enfrentarse sin contar con experiencia previa propia o de otros usuarios, ralentizando así la planificación del mismo y requiriendo de un mayor esfuerzo técnico.

### 7.1.2 Entorno en movimiento

Se considera importante destacar que a lo largo de este proyecto, además de los cambios comunes debido a nuevos conocimientos y mejores alternativas, se produjeron gran cantidad de cambios en las plataformas, librerías y herramientas empleadas.

Esto se debe principalmente a que el entorno de las HoloLens 2 es novedoso y en constante movimiento, lo que requiere de una especial atención a las novedades de los desarrolladores y una capacidad de adaptación al cambio debido a las modificaciones obligadas en los requisitos e implementaciones de librerías y herramientas.

### 7.1.3 Personalización plena y problemas asociados

Lo más destacable del desarrollo de esta herramienta son los problemas asociados a uno de los objetivos principales del proyecto, consistente en la capacidad de adaptación de la aplicación a cualquier arquitectura de [CPD](#) definida. Es decir, una personalización plena sobre los modelos a representar, que sea capaz de representar diferentes entornos sin modificar el código de la herramienta. Unity no está pensado para un desarrollo dinámico de estas características, pues la mayor parte de los proyectos cuentan con unos objetos y relaciones entre ellos pre-establecidos. Conseguir obtener enlaces similares entre objetos creados de forma dinámica supuso todo un reto para el equipo técnico, que necesitó de varias jornadas de trabajo para obtener los conocimientos necesarios para establecer un sistema de referencias dinámico a la par que eficiente, cuestión primordial debido a la naturaleza de las HoloLens. Es por ello que se remarcaba la capacidad del programa de establecer referencias entre los distintos componentes e interfaces en tiempo de ejecución sin comprometer a la estabilidad del sistema.

## 7.2 Pruebas

Las diferentes pruebas aplicadas a este proyecto no requerían de un sistema automatizado debido a la naturaleza del mismo, sino que se realizaban comprobaciones directas tras el desarrollo del código. La plataforma Unity permite imitar el comportamiento de las HoloLens hasta cierto punto, tal y como se describe en la sección 7.1, y dado que no requiere de una compilación completa de varios minutos sino que se tardan apenas segundos, las pruebas se realizaban al finalizar el objetivo buscado y se anotaba en el cuaderno Kanban (sección 4.1.2) los errores detectados. Sin embargo, podemos distinguir dos tipos de pruebas que requerían de un proceso más avanzado: pruebas en funcionalidad y pruebas en fluidez.

### 7.2.1 Pruebas en funcionalidad

Si bien Unity replica el comportamiento de las HoloLens, la propia naturaleza de las mismas (movimiento e interacción con las manos, colocación de los cascos...) obligaba cada determinado tiempo a realizar una compilación completa (sección A.1) y realizar las pruebas manuales directamente desde las propias HoloLens. De esta manera se obtenían resultados fiables en cuanto a las funcionalidades desarrolladas, y que conllevaron cambios en el diseño y desarrollo de interfaces y funciones auxiliares.

### 7.2.2 Pruebas en fluidez

Por otro lado, la plataforma Unity se ejecuta directamente en el PC, por lo que las características hardware no son equivalentes a ejecutar la aplicación en las HoloLens. Por ello, las pruebas de fluidez en la aplicación, muy importantes para permitir al usuario una ejecución cómoda del programa, se realizaban cada determinado tiempo con el objetivo de obtener feedback directo de las capacidades de procesamiento de las HoloLens. Eso conllevó, por ejemplo, al cambio de modelos 3D en los primeros compases del desarrollo (sección 6.4), ya que desde el equipo la aplicación se ejecutaba sin problemas pero las limitaciones técnicas de las HoloLens reducía considerablemente la comodidad del usuario, al ralentizar y bloquear el programa.

### 7.2.3 Visual Profiler

Para la comprobación del uso de recursos hardware por parte de la aplicación activa, la librería MRTK ofrece la herramienta Visual Profiler [64], la cual se puede apreciar en la figura 7.2.

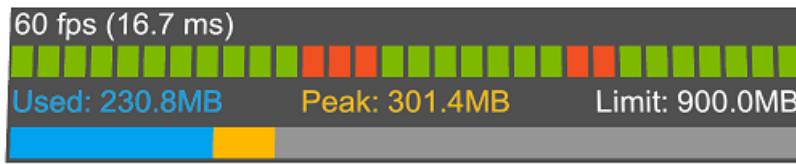


Figura 7.2: Captura de la herramienta Visual Profiler en la que se aprecian los recursos que monitoriza

Esta herramienta permite monitorizar en tiempo real los recursos empleados en la ejecución del programa, tales como CPU o RAM, y mostrar la tasa de **Fotogramas por segundo (FPS)**. Es utilizada para depurar posibles errores que afecten a la fluidez de la aplicación, ya que según interactuamos con la misma, el Profiler muestra los cambios de uso de recurso en su propia interfaz (figura 7.3).



Figura 7.3: Captura de VirManDC en la que se aprecia el Visual Profiler

De esta manera, se pudieron detectar los problemas mencionados en la sección 7.2.2, ya que como se aprecia en la figura 7.4 la visualización de los modelos avanzados consumía una gran cantidad de recursos, reduciendo así la fluidez del programa y generando malestar visual.

Tras corregir los modelos y mejorar el uso de recursos del programa a lo largo de diversas iteraciones, se logró una alta eficiencia de la aplicación, que se reflejaba en el Visual Profiler con un código verde prácticamente continuo.

Únicamente en el caso de operaciones realmente costosas, como la lectura y carga directa de los modelos 3D a partir de los ficheros XML, el Visual Profiler muestra un elevado uso de recursos. En este caso, aunque se ha maximizado la eficiencia del código, al ser una operación costosa no se puede reducir el pico de uso que se aprecia en la figura 7.5. Sin embargo, para evitar producir problemas al usuario, se le informa de la pausa visual mediante un Indicador de progreso [65], que aparece en la pantalla de carga descrita en la sección 5.2.1.



Figura 7.4: Captura de VirManDC en la que se aprecia el Visual Profiler mostrando un elevado uso de recursos

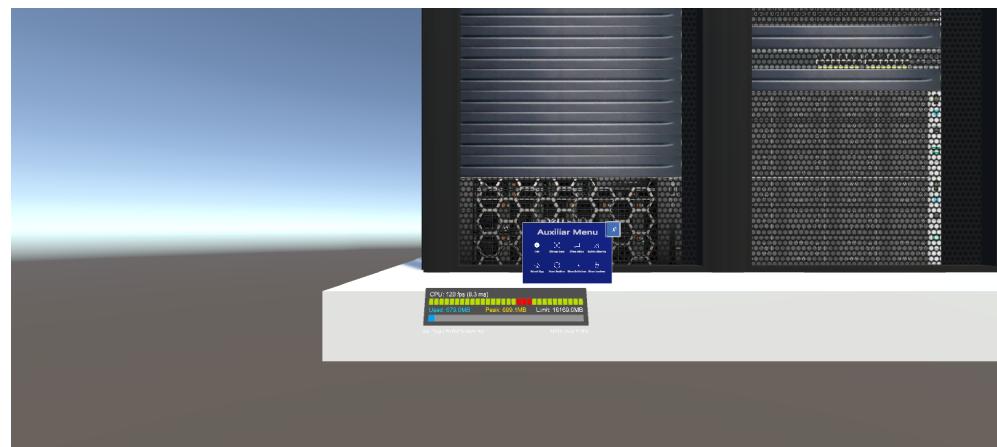


Figura 7.5: Captura de VirManDC con el Visual Profiler mostrando un pico en el uso de recursos



## Capítulo 8

# Conclusiones y trabajo futuro

---

**E**n este capítulo se reflexiona sobre las conclusiones obtenidas al finalizar este proyecto, tanto generales como personales.

## 8.1 Conclusiones

Se ha decidido separar este capítulo en dos bloques principales, uno con las conclusiones generales del proyecto en base a lo establecido en los objetivos del mismo, y otro con una reflexión más personal.

### 8.1.1 Conclusiones generales

Se puede afirmar que lo propuesto en los objetivos de este trabajo se ha conseguido, pues se ha logrado crear una aplicación que sea capaz de monitorizar e interactuar en tiempo real con un centro de procesamiento de datos mediante el uso de una herramienta de visualización inmersiva, además de ser capaz de adaptarse a otros CPDs sin tener que modificar el código.

En cuanto a los requisitos secundarios planteados para la herramienta:

- Se emplea un entorno virtual y/o holográfico para un caso de uso útil en el sector de las tecnologías de la información.
- La herramienta desarrollada emplea un entorno holográfico/virtual para su ejecución, demostrando la capacidad de una tecnología puntera.
- La plataforma se puede emplear perfectamente como sistema de soporte y formación para el aprendizaje del personal técnico en las tareas de monitorización de un CPD, con unas visuales realistas y comportamiento similar.

- La herramienta supone una alternativa no presencial para un entorno de trabajo en el que suele ser necesario estar presente, sin eliminar los procesos más importantes.

### 8.1.2 Conclusiones personales

Desde el punto de vista personal, la realización de este trabajo ha permitido experimentar las tareas propias de un ingeniero informático en primera persona. Ya sea trabajando en el diseño y la planificación, pasando por la obtención de requisitos y la correcta utilización de los recursos de programación, hasta el desafío de enfrentarse a tecnologías desconocidas y sin apenas documentación existente.

Por citar algunas de las asignaturas incluidas en el Máster Universitario de Enxeñaría Informática (MUEI) que han permitido adquirir los conocimientos necesarios para la correcta realización de este TFM, se incluyen las siguientes:

- **Análisis de sistemas de información (ASI):** Gracias a esta asignatura fue sencillo encontrar un enfoque metodológico a la hora de extraer los requisitos funcionales y no funcionales del problema planteado.
- **Diseño de sistemas de información (DSI):** Piedra angular de la implementación de este proyecto fueron los conocimientos adquiridos en esta asignatura, tales como patrones, técnicas de diseño y desarrollo de metodología ingenieril, prototipados, desarrollo dinámico, eficiencia, etc.
- **Informática como servicio (IS):** Los conocimientos relacionados con el Big Data, tales como servidores o computación en la nube, sirvieron para familiarizarse más rápidamente con los conceptos de funcionamiento del CPD sobre el que se trabajó.
- **Interacción, gráficos y multimedia (IGM):** Esta asignatura facilitó el entendimiento y manejo de entornos virtuales y holográficos, y otorgó los conocimientos y herramientas básicas para el desarrollo y la optimización de los diversos componentes gráficos (texturas, profundidad, cálculos complejos del motor gráfico, etc.)
- **Arquitecturas y plataformas móviles (APM):** El componente móvil de este proyecto permite aprovechar las metodologías y patrones orientados a las plataformas móviles que se enseñan en esta asignatura.
- **Dirección de proyectos (DP):** Una correcta planificación del proyecto fue clave para su finalización, pues estableció la base para un desarrollo estable y la facilidad en la gestión de problemas.

Para finalizar, me siento orgulloso y contento de haber podido finalizar este trabajo, con el gran esfuerzo que conllevó, con tantas lecciones aprendidas, y con las oportunidades que me brinda de cara al futuro.

## 8.2 Trabajo futuro

Si bien se entiende que se ha logrado alcanzar los objetivos definidos para el alcance de este proyecto, en esta sección se proponen los posibles pasos a seguir o características de la herramienta que tienen margen de mejora:

- Implementación avanzada de Realidad Mixta, interactuando directamente con los servidores en el mundo real. Si bien con el enfoque actual se elimina la problemática de los problemas de salud relacionados con una estancia excesiva en el interior de un CPD, no se exprimen las características de las HoloLens al 100%. Un posible paso posterior podría ser el añadir un sistema de lectura de QRs a los servidores físicos, abriendo de ese modo la interfaz principal de monitorización del servidor.
- Crear un programa externo que mejore la creación de la plantilla XML. Si bien se escogió un formato cómodo para ser usado sin tener un conocimiento avanzado, completar todo un XML para un CPD de grandes dimensiones puede llegar a ser una tarea tediosa, que se podría agilizar con un ligero programa en React o alguna tecnología similar que generase el fichero XML a partir de inputs visuales.
- Obtener información de otras fuentes de datos. Si bien hasta este momento sólo se muestra la información obtenida a través de Zabbix, se podrían implementar conexiones a más APIs, como por ejemplo las de los InRow, la de las interfaces de gestión de los propios servidores (idrac) u otros sistemas de monitorización adicionales (como el sistema de sensórica basado en IoT desarrollado por el investigador Alejandro Manuel Mosteiro Vázquez [66]) con el objetivo de mejorar el conocimiento en tiempo real del usuario.
- Inclusión de SOPs. Debido a las características propias de las HoloLens 2, es relativamente sencillo añadir guías visuales para ampliar los mecanismos de aprendizaje de los usuarios.



# **Apéndices**



## Apéndice A

# Guía de usuario

---

**E**n este apartado se incluye como anexo una guía de configuración básica de la herramienta, destinada al usuario de la misma. Además de explicar los procesos de compilación del programa, es de utilidad para todo aquel usuario que necesite modificar los ajustes de representación gráfica de la herramienta o de las credenciales de Zabbix.

## A.1 Instalación del programa

El proceso de instalación es necesario la primera vez que se quiera usar la aplicación en las HoloLens, y cada vez que se modifique el proyecto en Unity (al añadir gráficos o cambiar el código, por ejemplo). Este proceso no es necesario para guardar los ajustes que se hacen en los archivos de configuración.

Para poder realizar la instalación es necesario tener acceso al repositorio del código del programa, tener instalado Unity versión mínima 2018.4 LTS y tener instalado Visual Studio 2019/2020. Para una guía más detallada, consulte la documentación oficial [67].

Tras descargar el código de la aplicación, se deberá abrir como proyecto de Unity desde el menú del Unity Hub.

Una vez abierto el proyecto, se realizará la primera compilación (o Build<sup>1</sup>) con la configuración de la figura A.1. Es posible que falten librerías de Unity si no están preinstaladas, por lo que se deberán seguir los pasos definidos en la documentación oficial anteriormente mencionada.

---

<sup>1</sup> En Unity, el proceso de Build es la compilación del programa en un ejecutable listo para ser lanzado por la plataforma elegida como destino. En el caso de las HoloLens, es necesario un proceso de dos compilaciones, una desde Unity y otras desde Visual Studio

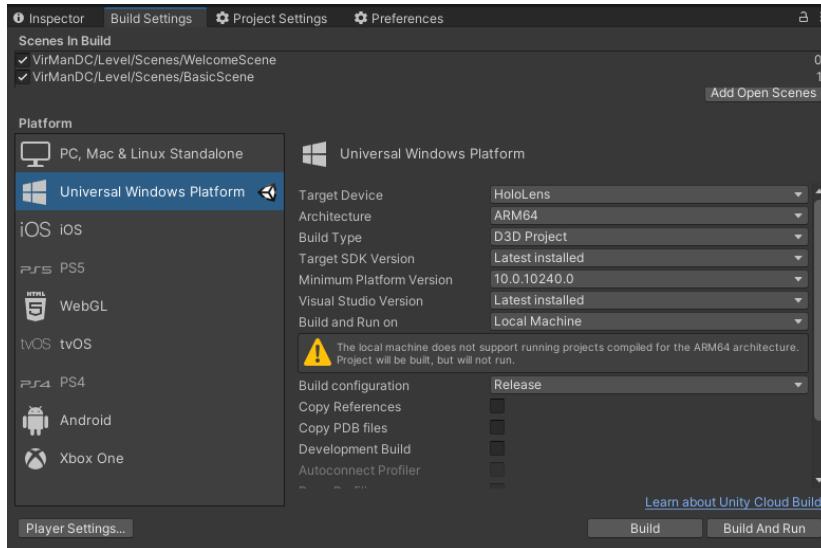


Figura A.1: Ajustes para la primera compilación del proyecto desde Unity

Tras compilar, se abrirá la Build desde Visual Studio 2020 (Importante: no abrir el proyecto como tal, sino el resultado de la compilación, que estará en una carpeta aparte).

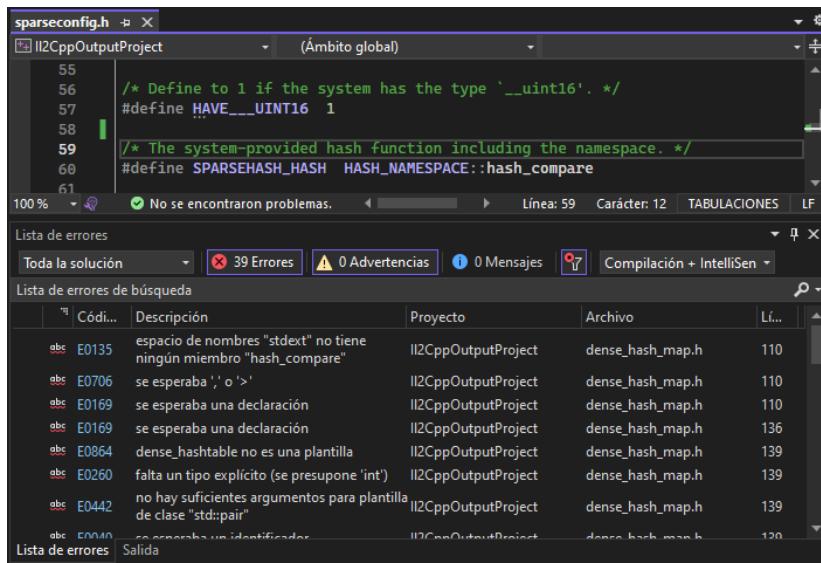


Figura A.2: Segunda compilación del proyecto desde Visual Studio, aparece el error mencionado

Tras conectar las gafas HoloLens al PC, se compilará el proyecto desde Visual Studio. Si aparece el error mostrado en la figura A.2, es necesario realizar un paso adicional [68], y tras ello debería mostrarse una salida como la de la figura A.3.

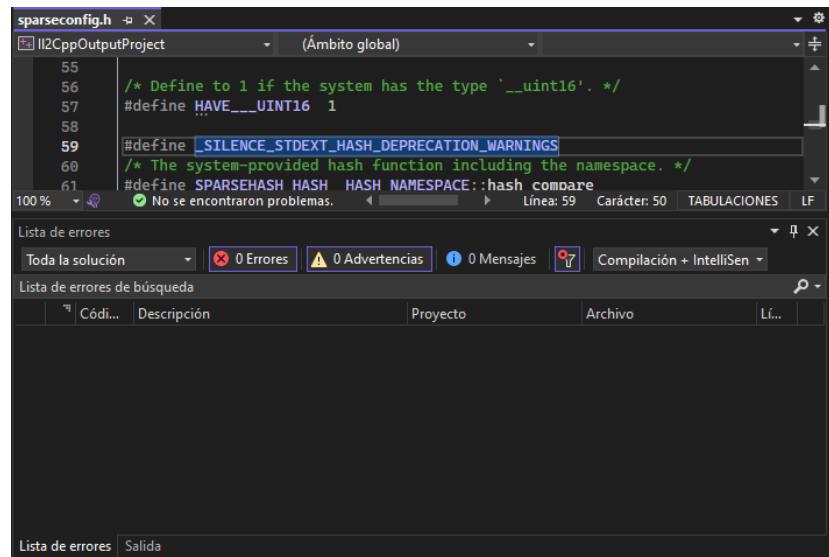


Figura A.3: Segunda compilación del proyecto desde Visual Studio, desaparece el error tras aplicar la solución

Una vez termine la compilación, el dispositivo HoloLens debería abrir directamente la aplicación. Ya se puede retirar el cable conectado al PC o cerrar la aplicación, pues ésta ya estará instalada en las HoloLens.

## A.2 Configuración inicial

Una vez se haya instalado la aplicación, se deberá crear una carpeta que contenga los ficheros de configuración necesarios para conectar a un servidor Zabbix y mostrar los modelos 3D. Se puede encontrar una muestra de dichos ficheros junto con el código del programa, preparada para mostrar el CPD del CITIC. Sin embargo, si no se poseen las credenciales, no se podrá conectar al servidor Zabbix del mismo.

La carpeta debe contener lo siguiente:

- Archivo VMDCConfiguration.config
- Archivo VMDCConfigurationDefault.config
- Carpeta ConfigurationFiles, con el contenido:
  - Architecture\_CITIC\_CPD.xml
  - Architecture\_CITIC\_CPD\_little.xml
  - IndicatorInterfacesModels.xml
  - KeyValueModels.xml
  - IndicatorsPanelModels.xml
  - RackDataModels.xml
  - ZabbixScripts.xml

Esta carpeta deberá ser copiada en la siguiente ruta ([A.4](#)) dentro de las HoloLens (figura A.5):

<sup>1</sup> Users Folders \ LocalAppData \ VirManDC App Name \ LocalState

Figura A.4: Ruta del sistema de archivos de HoloLens en la que copiar la carpeta de configuración

## A.3 Configuración ficheros XML

Estos ficheros serán los encargados de representar correctamente tanto los modelos gráficos como las interfaces de los datos de Zabbix. Es recomendable dejar los ficheros por defecto si no se van a utilizar, para evitar errores debido a una incorrecta interpretación de los XML. En la pantalla inicial, en la interfaz de error, se muestra el tipo de error encontrado si la carga de los ficheros XML no es la correcta.

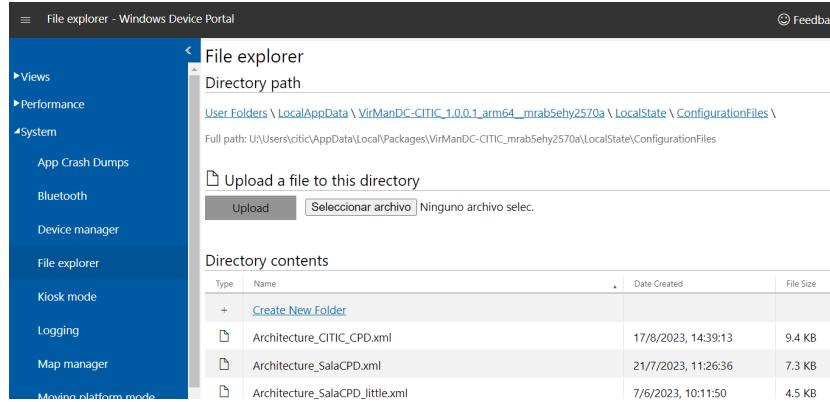


Figura A.5: Captura de la interfaz web de las HoloLens 2, se muestra la ruta de instalación de los archivos de configuración

### A.3.1 Fichero Architecture\_CITIC\_CPD.xml

Este fichero será la fuente principal de la carga de los modelos 3D. Como se puede apreciar en la figura A.6, se comienza definiendo el atributo *PlaceDescription*. En estos campos se define la posición inicial y la escala en los dos ejes horizontales de la plataforma sobre la que situarán los modelos 3D. Es necesario ajustarlos si se modifica la localización de dichos modelos.

```

1  <?xml version="1.0" encoding="utf-8"?>
2  | <!-- Last edition: 30/08/2023 17:4w0 -->
3  |   <Architecture>
4  |     <PlaceDescription posX="-0.4" scaleX="7.2" posZ="-0.7" scaleZ="3.6" centerX="1.5" centerZ="1.2"/>
5  |   <RacksDefinition>
6  |     <Rack rackID="101" name="Rack 1.1" model="BasicModel_Rack" graphicsModel="LP">
7  |       <Transform>
8  |         <posX>0</posX>
9  |         <posY>0.85</posY>
10 |         <posZ>0</posZ>
11 |         <rotation>0</rotation>
12 |       </Transform>

```

Figura A.6: Sección del fichero Architecture\_CITIC\_CPD.xml en la que se aprecia el atributo PlaceDescription

En el siguiente atributo, *RacksDefinition*, se situarán de forma anidada los racks, rodeados por las etiquetas *Rack*. Dichas etiquetas tendrán los siguientes atributos:

- *rackID*: ID único que puede ser de utilidad para identificar a cada modelo desde la interfaz de Unity (no necesario, conveniente para debuggear la aplicación).
- *name*: el nombre con el que aparecerá el modelo en la aplicación (visible en las etiquetas).
- *model*: el nombre del archivo del modelo 3D. Prefijo que junto con el atributo *name* conforman el nombre del modelo que se cargará desde la carpeta correspondiente (ver sección 6.2.1).

- *graphicsModel*: el tipo de modelo 3D. Sufijo que junto con el atributo *name* conforman el nombre del modelo que se cargará desde la carpeta correspondiente (ver sección 6.2.1).

A su vez, dentro de cada rack se situarán dos etiquetas anidadas, una llamada *Transform* que contendrá la información necesaria para situar el modelo en el espacio (posición y rotación), y otra llamada *SlotsRack* que contendrán de forma anidada los servidores físicos (en forma de Slot).

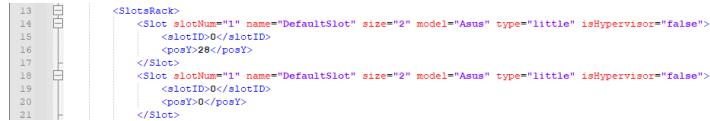


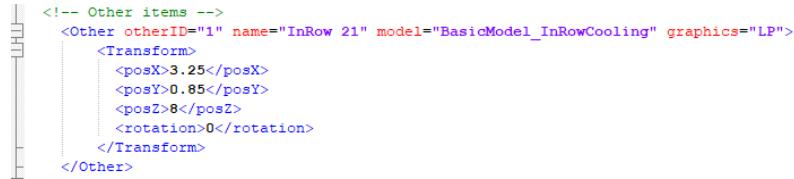
Figura A.7: Sección del fichero Architecture\_CITIC\_CPD.xml en la que se aprecia el atributo SlotsRack

Como se aprecia en la figura A.7, cada etiqueta *Slot* contiene los siguientes atributos:

- *slotNum*: ID único que puede ser de utilidad para identificar a cada modelo desde la interfaz de Unity (no necesario, conveniente para debuggear la aplicación).
- *name*: el nombre con el que aparecerá el modelo en la aplicación (visible en el título de la interfaz principal del servidor, 5.4).
- *size*: tamaño vertical que ocupará el frontal del slot. El frontal se autoajusta, pero es necesario comprobar que no se solapa con otros slots en posiciones cercanas.
- *model*: el nombre de la imagen frontal que se cargará desde la carpeta correspondiente (ver sección 6.2.1).
- *type*: el tipo de interfaz que se cargará. Referencia a A.3.3.
- *isHypervisor*: define si el slot contiene un servidor hypervisor de Zabbix (lo que conlleva llamadas a la API para consultar las máquinas virtuales que hostea).

Además, tendrá en su interior las etiquetas *slotID*, que define el identificador de Zabbix empleado (se puede utilizar el HostID de Zabbix o su IP, pero esta última debe ser única. El slotID debe ser 0 si no se desea monitorizar); y *posY*, que definirá la altura o slot en el que se situará el servidor dentro del armario.

Por último, el archivo contendrá bajo la etiqueta *Other* los modelos 3D que no contengan información, es decir, los elementos gráficos auxiliares sin comportamiento asociado. Se configura de manera similar a los Racks, como se puede apreciar en la figura A.8.



```

<!-- Other items -->
<Other otherID="1" name="InRow 21" model="BasicModel_InRowCooling" graphics="LP">
    <Transform>
        <posX>3.25</posX>
        <posY>0.85</posY>
        <posZ>8</posZ>
        <rotation>0</rotation>
    </Transform>
</Other>

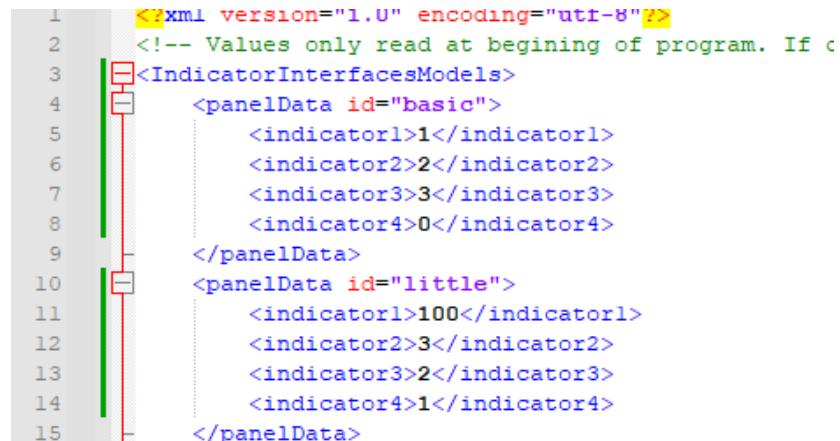
```

Figura A.8: Sección del fichero Architecture\_CITIC\_CPD.xml en la que se aprecia la etiqueta Others

### A.3.2 Fichero Architecture\_CITIC\_CPD\_little.xml

Versión limitada del fichero Architecture\_CITIC\_CPD.xml, con un contenido menor. Sigue exactamente la misma estructura que el anterior fichero, se utiliza para demostrar las capacidades de la aplicación para cargar diferentes arquitecturas de forma dinámica sin necesidad de reinstalar el programa.

### A.3.3 Fichero IndicatorInterfacesModels.xml



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- Values only read at begining of program. If c
3  <IndicatorInterfacesModels>
4      <panelData id="basic">
5          <indicator1>1</indicator1>
6          <indicator2>2</indicator2>
7          <indicator3>3</indicator3>
8          <indicator4>0</indicator4>
9      </panelData>
10     <panelData id="little">
11         <indicator1>100</indicator1>
12         <indicator2>3</indicator2>
13         <indicator3>2</indicator3>
14         <indicator4>1</indicator4>
15     </panelData>

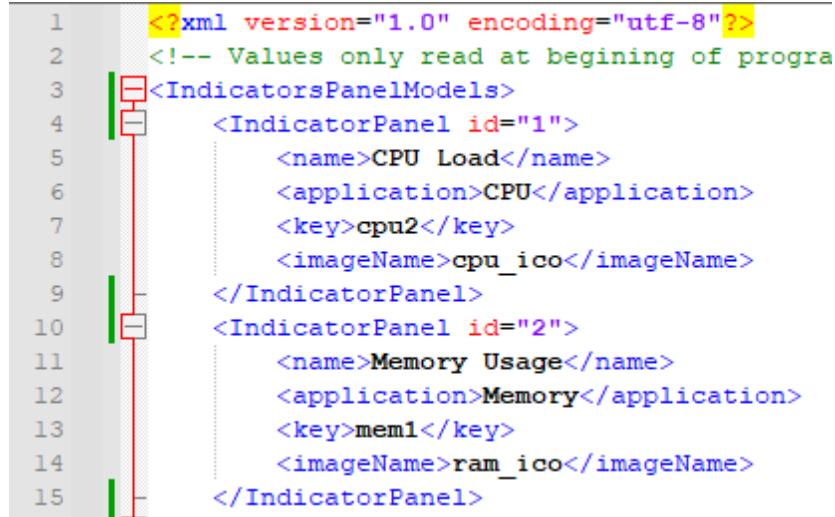
```

Figura A.9: Sección del fichero IndicatorInterfacesModels.xml

En este fichero se definen los modelos del panel principal que se mostrará al interactuar con los servidores monitorizados. Bajo la etiqueta *IndicatorInterfacesModels* se agrupan de forma anidada los modelos, a partir de la etiqueta *panelData*, que contiene un atributo llamado *id* para identificar de forma única cada uno de ellos. En el interior de la etiqueta encontramos los siguientes atributos en forma de etiquetas individuales:

- *indicatorX*: cada uno de los indicadores que se mostrarán en la interfaz principal (esta etiqueta se repite hasta 4 veces). En caso de necesitar menos indicadores, se debe dejar su valor a 0. Es una referencia a A.3.4

#### A.3.4 Fichero IndicatorsPanelModels.xml



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- Values only read at begining of progra
3  |<IndicatorsPanelModels>
4  |  |<IndicatorPanel id="1">
5  |  |  |<name>CPU Load</name>
6  |  |  |<application>CPU</application>
7  |  |  |<key>cpu2</key>
8  |  |  |<imageName>cpu_ico</imageName>
9  |  |</IndicatorPanel>
10 |<IndicatorPanel id="2">
11 |  |<name>Memory Usage</name>
12 |  |<application>Memory</application>
13 |  |<key>mem1</key>
14 |  |<imageName>ram_ico</imageName>
15 |</IndicatorPanel>

```

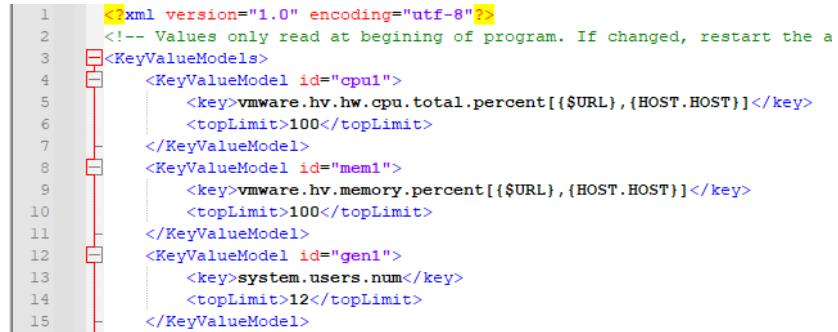
Figura A.10: Sección del fichero IndicatorsPanelModels.xml

En este fichero se definen los modelos de indicadores que se mostrarán en las interfaces principales de los servidores. Bajo la etiqueta *IndicatorsPanelModels* se agrupan de forma anidada los modelos de indicadores, a partir de la etiqueta *IndicatorPanel*, que contiene un atributo llamado *id* para identificar de forma única cada uno de ellos. En el interior de la etiqueta encontramos los siguientes atributos en forma de etiquetas individuales:

- *name*: el nombre con el que aparecerá el indicador (se muestra en el título de la interfaz de datos de Zabbix).
- *application*: apartado de Zabbix a partir del cual se extraen los datos del indicador, [6.6.4](#). También se utiliza para mostrar en el título de la interfaz de datos de Zabbix.
- *key*: identificador del componente a partir del cual se extrae la información. Es una referencia a [A.3.5](#).
- *imageName*: nombre de la imagen que representa el indicador, que debe ir en la sección correspondiente ([6.2.1](#)).

#### A.3.5 Fichero KeyValueModels.xml

En este fichero se definen los modelos de los valores clave que se mostrarán en los indicadores de las interfaces principales de los servidores. Bajo la etiqueta *KeyValueModels* se agrupan de forma anidada los modelos de claves, a partir de la etiqueta *KeyValueModel*, que



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- Values only read at begining of program. If changed, restart the a
3  <KeyValueModels>
4      <KeyValueModel id="cpul">
5          <key>vmware.hv.hw.cpu.total.percent[$URL],{HOST.HOST}</key>
6          <topLimit>100</topLimit>
7      </KeyValueModel>
8      <KeyValueModel id="mem1">
9          <key>vmware.hv.memory.percent[$URL],{HOST.HOST}</key>
10         <topLimit>100</topLimit>
11     </KeyValueModel>
12     <KeyValueModel id="gen1">
13         <key>system.users.num</key>
14         <topLimit>12</topLimit>
15     </KeyValueModel>

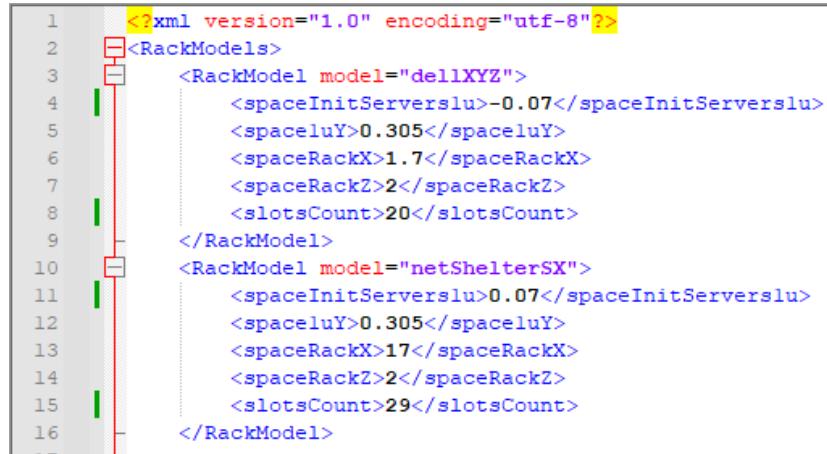
```

Figura A.11: Sección del fichero KeyValueModels.xml

contiene un atributo llamado *id* para identificar de forma única cada uno de ellos. En el interior de la etiqueta encontramos los siguientes atributos en forma de etiquetas individuales:

- *key*: valor de la clave que se utiliza, junto con el nombre de la aplicación, para extraer datos de Zabbix (6.6.4)
- *topLimit*: valor máximo utilizado para el cálculo del porcentaje del indicador

#### A.3.6 Fichero RackDataModels.xml



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <RackModels>
3      <RackModel model="dellXYZ">
4          <spaceInitServerslu>-0.07</spaceInitServerslu>
5          <spaceeluY>0.305</spaceeluY>
6          <spaceRackX>1.7</spaceRackX>
7          <spaceRackZ>2</spaceRackZ>
8          <slotsCount>20</slotsCount>
9      </RackModel>
10     <RackModel model="netShelterSX">
11         <spaceInitServerslu>0.07</spaceInitServerslu>
12         <spaceeluY>0.305</spaceeluY>
13         <spaceRackX>17</spaceRackX>
14         <spaceRackZ>2</spaceRackZ>
15         <slotsCount>29</slotsCount>
16     </RackModel>
17

```

Figura A.12: Sección del fichero RackDataModels.xml

En este fichero se definen las características de los modelos 3D utilizados en la aplicación. Bajo la etiqueta *RackModels* se agrupan de forma anidada los modelos 3D, a partir de la etiqueta *RackModel*, que contiene un atributo llamado *model* para identificar de forma única cada uno de ellos. En el interior de la etiqueta encontramos los siguientes atributos en forma de etiquetas individuales:

- *spaceInitServers1u*: espacio inicial para instanciar los servidores físicos (slots), ya que la altura a la que se empiezan a colocar los servidores no es justo 0.
- *space1uY*: longitud de espacio entre dos slots, para instanciar correctamente los servidores (evitar que se solapen).
- *spaceRackX*: valor por el que se multiplica la posición del modelo en el eje X definida en la Arquitectura para un ajuste más preciso.
- *spaceRackZ*: valor por el que se multiplica la posición del modelo en el eje Z definida en la Arquitectura para un ajuste más preciso.
- *slotsCount*: número de slots que posee el armario para contener servidores.

### A.3.7 Fichero ZabbixScripts.xml

```

1  <?xml version="1.0" encoding="utf-8"?>
2  <!-- Values only read at begining of program. If changed, r
3  <ZabbixScriptList>
4  <ZabbixScript id="1">
5      <ShowName>Ping</ShowName>
6  </ZabbixScript>
7  <ZabbixScript id="2">
8      <ShowName>Comprobar Sistema Operativo</ShowName>
9  </ZabbixScript>
10 <ZabbixScript id="3">
11     <ShowName>Datastore Usage</ShowName>
12 </ZabbixScript>
13 </ZabbixScriptList>

```

Figura A.13: Sección del fichero ZabbixScripts.xml

En este fichero se definen las referencias a los scripts de Zabbix utilizados en la aplicación. Bajo la etiqueta *ZabbixScriptList* se agrupan de forma anidada las referencias a los scripts, a partir de la etiqueta *ZabbixScript*, que contiene un atributo llamado *id* para identificar de forma única cada uno de ellos. Este ID se corresponde al ID interno de los scripts que se encuentran en la interfaz del servidor de Zabbix. En el interior de la etiqueta se encuentra el atributo *ShowName* en forma de etiqueta individual, que será el nombre mostrado en la interfaz de ejecución de scripts de Zabbix.

## A.4 Configuración ficheros JSON

Estos ficheros serán los encargados de establecer la conexión con el servidor de Zabbix.

```
[{"ipServer": "uatu.citic.udc.es",  
 "urlZabbixAPI": "api_jsonrpc.php",  
 "urlConfigurationFiles": "",  
 "encryptedUser": [REDACTED],  
 "encryptedPass": [REDACTED]}]
```

Figura A.14: Sección del fichero Configuration.json

#### A.4.1 Fichero Configuration.json

Este fichero es modificable también por las propias HoloLens, por lo que su contenido puede perderse tras editar los ajustes directamente desde la aplicación.

Se ha activado el permiso de escritura al usuario en caso de que no sea posible desde la aplicación.

Contiene los siguientes campos:

- *ipServer*: IP del servidor de Zabbix.
- *urlZabbixAPI*: URL dentro del servidor Zabbix al que apunta la API.
- *urlConfigurationFiles*: URL de configuración de ficheros, actualmente sin uso.
- *encryptedUser* y *encryptedPass*: credenciales de usuario de Zabbix encriptadas.

#### A.4.2 Fichero DefaultConfiguration.json

```
{"ipServer": "uatu.citic.udc.es",  
 "urlZabbixAPI" : "api_jsonrpc.php",  
 "urlConfigurationFiles" : ""}
```

Figura A.15: Sección del fichero DefaultConfiguration.json

Este fichero no es editable desde la aplicación, y contiene los valores por defecto que carga la aplicación. Se deberían ajustar según sea necesario, para en caso de modificación intencional desde la aplicación de los parámetros de configuración, poder volver cómodamente a unos ajustes funcionales. Es importante destacar que las credenciales no se guardan. Contiene los siguientes campos:

- *ipServer*: IP del servidor de Zabbix.
- *urlZabbixAPI*: URL dentro del servidor Zabbix al que apunta la API.
- *urlConfigurationFiles*: URL de configuración de ficheros, actualmente sin uso.

# **Lista de acrónimos**

---

**API** Application Programming Interfaces. 18, 28, 36, 43, 70, 89

**AES** Estándar Avanzado de Encriptación. 61

**CPD** Centro de Procesamiento de Datos. 1, 2, 7, 17, 40, 56, 82, 87

**FPS** Fotogramas por segundo. 84

**IA** Inteligencia Artificial. 7

**IAR** Industrial Augmented Reality. 11, V

**JSON** JavaScript Object Notation. 60, 70, 71

**LTS** Long Term Support. 68

**ML** Machine Learning. 7

**MVP** Producto Mínimo Viable. 28

**NAS** Network Attached Storage. 8

**PNG** Portable Network Graphics. 65

**RA** Realidad Aumentada. 5, 6

**RM** Realidad Mixta. 5, 6, 14

**RV** Realidad Virtual. 5, 14

**SAI** Sistema de Alimentación Ininterrumpida. 8, 40

**SOP** Standard Operating Procedure. 8, 9, 89

**TFG** Trabajo Fin de Grado. 11

**TFM** Trabajo Fin de Máster. 1

# Bibliografía

---

- [1] M. Arregoces and M. Portolani, *Data center fundamentals*. Cisco Press, 2003.
- [2] S. V. Patankar, “Airflow and cooling in a data center,” *Journal of Heat Transfer*. Jul 2010, 132(7): 073001, 2010.
- [3] J. Fink, *Impact of High Density Hot Aisles on IT Personnel Work Conditions*. APC, 2015. [En línea]. Disponible en: <https://it-resource.schneider-electric.com/white-papers/wp-123-impact-of-high-density-hot-aisles-on-it-personnel-work-conditions>
- [4] B. Wang, D. Schlagwein, D. Cecez-Kecmanovic, and M. C. Cahalane, “Editorial: Beyond the factory paradigm: Digital nomadism and the digital future(s) of knowledge work post-covid-19,” 2020. [En línea]. Disponible en: <https://aisel.aisnet.org/jais/vol21/iss6/10/>
- [5] S. Deffeyes, “Mobile augmented reality in the data center,” 2011. [En línea]. Disponible en: <https://ieeexplore.ieee.org/abstract/document/6032768>
- [6] P. MILGRAM and F. KISHINO, “A taxonomy of mixed reality visual displays,” 1994. [En línea]. Disponible en: [https://search.ieice.org/bin/summary.php?id=e77-d\\_12\\_1321](https://search.ieice.org/bin/summary.php?id=e77-d_12_1321)
- [7] Beyond.pl, “Smart hands with microsoft hololens 2.0,” 2021. [En línea]. Disponible en: <https://www.beyond.pl/en/services/smart-hands-with-microsoft-hololens/>
- [8] “Microsoft usa hololens para auditar los centros de datos de forma remota,” <https://www.profesionalreview.com/2021/04/29/microsoft-hololens-centros-datos-remota/>, 2021.
- [9] B. Peng, S. Hung, G. Hung, Y. Pan, F. Lo, and M. Tsai, “Hololens 2 data center demo,” 2022. [En línea]. Disponible en: <https://www.youtube.com/watch?v=xZ1fibff1UY>
- [10] Inceptum, “Data center operations supported by augmented reality,” 2021. [En línea]. Disponible en: <https://www.inceptum-oss.com/data-center-operations-supported-by-augmented-reality/>

- [11] D. Ungureanua, F. Bogo, S. Galliani, P. Sama, X. Duan, C. Meekhof, J. Stühmer, T. J. Cashman, B. Tekin, J. L. Schönberger, P. Olszta, and M. Pollefeys, “Hololens 2 research mode as a tool for computer vision research,” 2008. [En línea]. Disponible en: <https://arxiv.org/abs/2008.11239v1>
- [12] Z. Makhataeva and H. A. Varol, “Augmented reality for robotics: A review,” 2020. [En línea]. Disponible en: <https://doi.org/10.3390/robotics9020021>
- [13] A. Palumbo, “Microsoft hololens 2 in medical and healthcare context: State of the art and future prospects,” 2022. [En línea]. Disponible en: <https://www.mdpi.com/1424-8220/22/20/7709>
- [14] M. Kaufeld, M. Mundt, S. Forst, and H. Hecht, “Optical see-through augmented reality can induce severe motion sickness,” 2022. [En línea]. Disponible en: <https://www.sciencedirect.com/science/article/pii/S0141938222001032>
- [15] H. zhi Hu, X. bo Feng, Z. wu Shao, M. Xie, S. Xu, X. huo Wu, and Z. wei Ye, “Application and prospect of mixed reality technology in medical field,” 2019. [En línea]. Disponible en: <https://link.springer.com/article/10.1007/s11596-019-1992-8>
- [16] D. Rivas González, “Sistema de ayuda a la supervisión de infraestructuras informáticas mediante técnicas de visualización inmersivas. integración con una herramienta de monitorización de código abierto,” 2017. [En línea]. Disponible en: <https://www.udc.es/es/tfe/traballo/?codigo=8742>
- [17] A. V. Balea, O. B. Novoa, P. F. Lamas, M. V. Montesinos, and T. M. F. Caramés, “A collaborative industrial augmented reality digital twin: Developing the future of shipyard 4.0,” 2022. [En línea]. Disponible en: [https://www.researchgate.net/publication/363020782\\_A\\_Collaborative\\_Industrial\\_Augmented\\_Reality\\_Digital\\_Twin\\_Developing\\_the\\_Future\\_of\\_Shipyard\\_40](https://www.researchgate.net/publication/363020782_A_Collaborative_Industrial_Augmented_Reality_Digital_Twin_Developing_the_Future_of_Shipyard_40)
- [18] Microsoft, “Hololens 2.” [En línea]. Disponible en: <https://www.microsoft.com/es-es/hololens>
- [19] Unity, “Web oficial de unity,” <https://unity.com/es>.
- [20] Epic Games, “Web oficial de unreal engine,” <https://www.unrealengine.com/>.
- [21] Microsoft, “Web oficial de mrtk,” <https://learn.microsoft.com/es-es/windows/mixed-reality/mrtk-unity/mrtk2>.
- [22] Zabbix, “Web oficial de zabbix,” <https://www.zabbix.com/>.

- [23] Nagios, “Web oficial de nagios,” <https://www.nagios.org/>.
- [24] Zenoss, “Web oficial de zenoss,” <https://www.zenoss.com/>.
- [25] Microsoft, “Web oficial de visual studio code,” <https://code.visualstudio.com/>.
- [26] ——, “Web oficial de visual studio 2022,” <https://visualstudio.microsoft.com/es/>.
- [27] Git, “Web oficial de git,” <https://git-scm.com/>.
- [28] Sourcetree, “Web oficial de sourcetree,” <https://www.sourcetreeapp.com/>.
- [29] Blender, “Web oficial de blender,” <https://www.blender.org/>.
- [30] FreeCAD, “Web oficial de freecad,” <https://www.freecad.org/>.
- [31] GIMP, “Web oficial de gimp,” <https://www.gimp.org/>.
- [32] C. L. Gómez, A. Álvarez García, and R. de las Heras del Dedo, *Métodos ágiles: Scrum, Kanban, Lean*, 1st ed. Anaya Multimedia, 2017.
- [33] H. Smith, *Scrum: The Ultimate Beginner’s Guide To Learn And Master Scrum Agile Framework*. CreateSpace Independent Publishing Platform, 2018.
- [34] K. Schwaber and J. Sutherland, “Scrum guide 2020,” <https://scrumguides.org/scrum-guide.html>.
- [35] J. Ries, *Beginner’s Guide to Mastering Agile Project Management with Scrum, Kanban, Scrumban, Lean, Six Sigma, and Extreme Programming*. Independently published, 2019.
- [36] Japan Management Association, *Kanban: Y “Just-in-Time” en Toyota*, 1st ed. TGP Hoshim, 1998.
- [37] A. Reddy, *Scrumban [R]Evolution, The: Getting the Most Out of Agile, Scrum, and Lean Kanban (Agile Software Development Series)*, 1st ed. Financial Times Prentice Hall, 2015.
- [38] Microsoft, “Hololens 2 development edition - microsoft store.” [En línea]. Disponible en: <https://www.microsoft.com/en-us/d/hololens-2-development-edition/92f64zpzzd4?activetab=pivot:overviewtab>
- [39] “BOE núm. 59, de 10 de marzo de 2023, páginas 36003 a 36043,” [https://www.boe.es/eli/es/res/2023/02/27/\(6\)](https://www.boe.es/eli/es/res/2023/02/27/(6)).
- [40] Microsoft, “Mrtk - direct manipulation with hands,” 2022. [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/design/direct-manipulation>

- [41] Unity, “Best practices for organizing your unity project.” [En línea]. Disponible en: <https://unity.com/how-to/organizing-your-project>
- [42] ——, “Resources class and folder.” [En línea]. Disponible en: <https://docs.unity3d.com/ScriptReference/Resources.html>
- [43] ——, “Unity documentation: Application.persistentdatapath.” [En línea]. Disponible en: <https://docs.unity3d.com/ScriptReference/Application-persistentDataPath.html>
- [44] Microsoft, “Common c# code conventions.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/coding-style/coding-conventions>
- [45] Google, “C# at google style guide.” [En línea]. Disponible en: <https://google.github.io/styleguide/csharp-style.html>
- [46] Refactoring Guru, “Design patterns.” [En línea]. Disponible en: <https://refactoring.guru/design-patterns>
- [47] Unity, “Important classes - monobehaviour.” [En línea]. Disponible en: <https://docs.unity.cn/ru/2020.2/Manual/class-MonoBehaviour.html>
- [48] A. X. García, “Repositorio oficial virmandc.” [En línea]. Disponible en: <https://github.com/AdrianXuizGarcia/VirManDC>
- [49] NIST, “Advanced encryption standard (aes),” 2001. [En línea]. Disponible en: <https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197.pdf>
- [50] StackOverflow, “Encrypting & decrypting a string in c#.” [En línea]. Disponible en: <https://stackoverflow.com/questions/10168240/encrypting-decrypting-a-string-in-c-sharp>
- [51] Microsoft, “Performance – mrtk2.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/performance/perf-getting-started?view=mrtkunity-2022-05>
- [52] VisioCafe, “Página oficial de visiocafe.” [En línea]. Disponible en: <https://www.visiocafe.com/>
- [53] Microsoft, “Visio file format reference.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/office/client-developer/visio/visio-file-format-reference>
- [54] Drawio, “Web oficial de draw.io.” [En línea]. Disponible en: <https://www.drawio.com/>
- [55] Ivanti, “Web oficial de pulse secure.” [En línea]. Disponible en: <https://www.ivanti.com/company/history/pulse-secure>

- [56] Zabbix, “Zabbix documentation: Api.” [En línea]. Disponible en: <https://www.zabbix.com/documentation/current/en/manual/api>
- [57] ECMA, “Ecma-404 the json data interchange syntax.” [En línea]. Disponible en: <https://www.ecma-international.org/publications-and-standards/standards/ecma-404/>
- [58] Unity, “Documentación oficial del serializador de unity.” [En línea]. Disponible en: <https://docs.unity3d.com/ScriptReference/Serializable.html>
- [59] Microsoft, “Mrtk - buttons.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/button>
- [60] ——, “Mrtk - near menu.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/near-menu>
- [61] ——, “Mrtk - hand menu.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/hand-menu>
- [62] ——, “Mrtk - sliders.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/sliders>
- [63] ——, “Mrtk - tooltip.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/tooltip>
- [64] ——, “Using the visual profiler - mrtk.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/diagnostics/using-visual-profiler>
- [65] ——, “Mrtk - progress indicator.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/mrtk-unity/mrtk2/features/ux-building-blocks/progress-indicator>
- [66] A. M. Mosteiro Vázquez, “Monitorización de infraestructuras ti incorporando sensórica basada en iot,” 2017. [En línea]. Disponible en: <https://www.udc.es/es/tfe/traballo/?codigo=29271>
- [67] Microsoft, “Build and deploy to the hololens.” [En línea]. Disponible en: <https://learn.microsoft.com/en-us/windows/mixed-reality/develop/unity/build-and-deploy-to-hololens>
- [68] Unity forum, “Workaround for building with il2cpp with visual studio 2022 17.4.” [En línea]. Disponible en: <https://forum.unity.com/threads/workaround-for-building-with-il2cpp-with-visual-studio-2022-17-4.1355570/>