

Apéndice A

Guía de usuario

EN 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 en [Learn Microsoft](#).

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 Build¹) 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.

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).

Tras conectar las gafas HoloLens al PC, se compilará el proyecto desde Visual Studio. Si aparece el error mostrado en la figura A.2, aplicar la solución de [este post en los foros de](#)

¹ 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

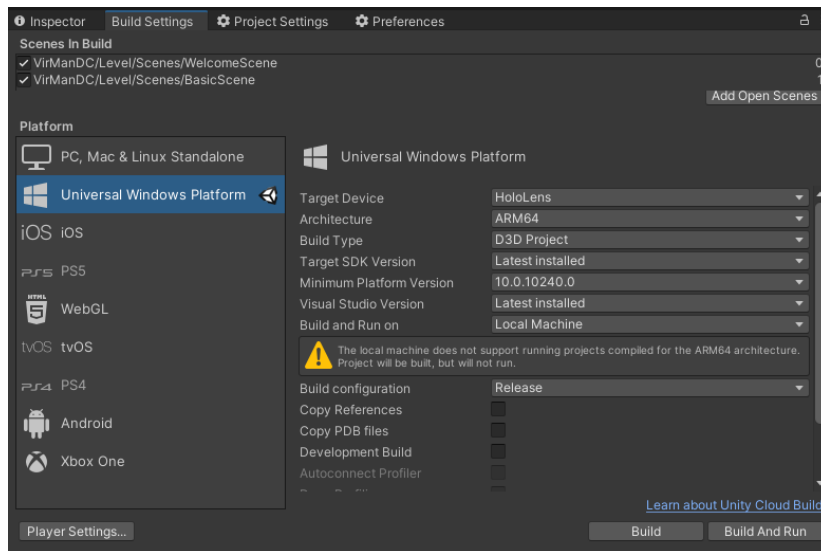


Figura A.1: Ajustes para la primera compilación del proyecto desde Unity

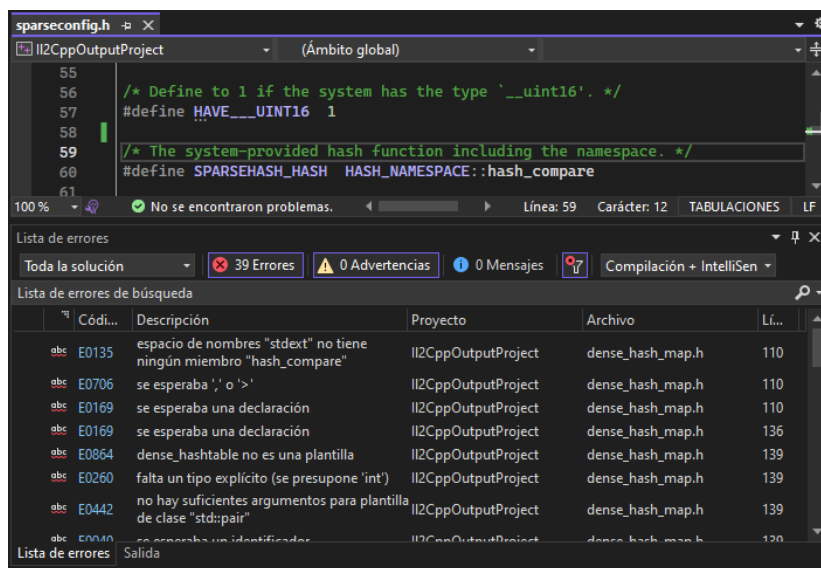


Figura A.2: Segunda compilación del proyecto desde Visual Studio, aparece el error mencionado

Unity, y debería mostrarse una salida como la de la figura A.3.

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.

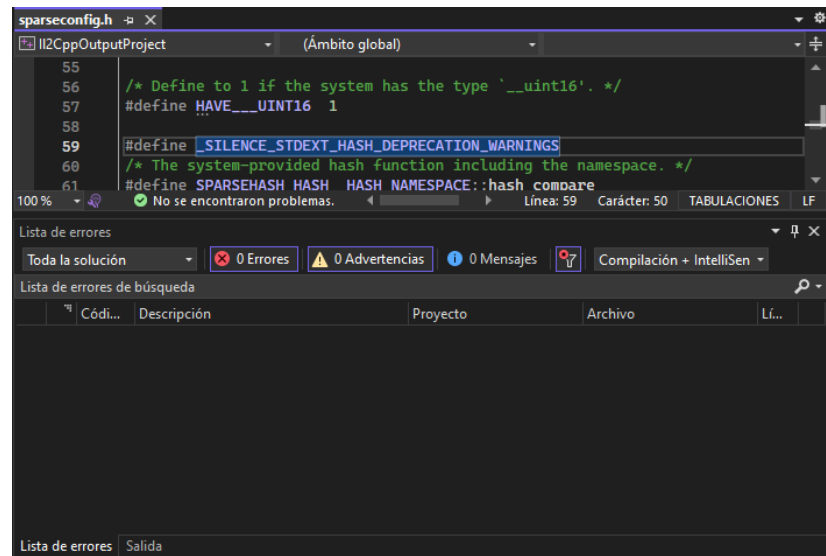


Figura A.3: Segunda compilación del proyecto desde Visual Studio, desaparece el error tras aplicar la solución

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 VMDCCConfiguration.config
- Archivo VMDCCConfigurationDefault.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):

1 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

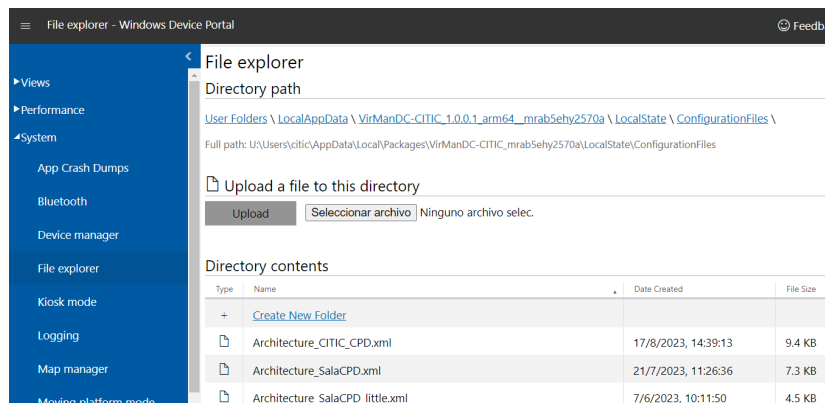


Figura A.5: Captura de la interfaz web de las HoloLens 2, se muestra la ruta de instalación de los archivos de configuración

A.3 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.

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.

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

```

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

- *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).

```

13 <SlotsRack>
14 <Slot slotNum="1" name="DefaultSlot" size="2" model="Asus" type="little" isHypervisor="false">
15 <slotID>0</slotID>
16 <posY>28</posY>
17 </Slot>
18 <Slot slotNum="1" name="DefaultSlot" size="2" model="Asus" type="little" isHypervisor="false">
19 <slotID>0</slotID>
20 <posY>0</posY>
21 </Slot>

```

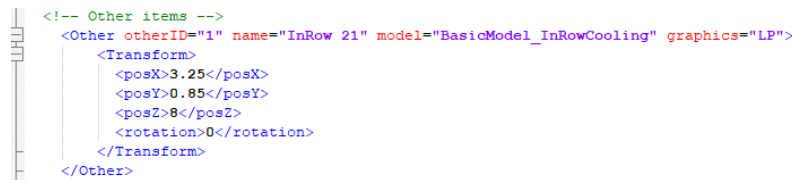
Figura A.7: Sección del fichero Architecture_CITIC_CPD.xml en la que se aprecia el atributo SlotsRack

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.



```
<!-- 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

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.

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

En este fichero se definen los modelos del panel principal (6.7.1) 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

```

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

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*: indentificador 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

```

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="cpu1">
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

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 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

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.


```

1  <?xml version="1.0" encoding="utf-8"?>
2  <RackModels>
3    <RackModel model="dellXYZ">
4      <spaceInitServerslu>-0.07</spaceInitServerslu>
5      <spaceY>0.305</spaceY>
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     <spaceY>0.305</spaceY>
13     <spaceRackX>17</spaceRackX>
14     <spaceRackZ>2</spaceRackZ>
15     <slotsCount>29</slotsCount>
16   </RackModel>
17

```

Figura A.12: Sección del fichero RackDataModels.xml

- *spaceY*: 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 beginning 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.

A.4.1 Fichero Configuration.json

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

Figura A.14: Sección del 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

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

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

Figura A.15: Sección del fichero DefaultConfiguration.json

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