



# **PROGRAMA COMANDOS PERSONALIZADOS PARA SISTEMA OPERATIVO**

**AA4. Estableciendo las  
bases de la integración de  
aplicaciones mediante la  
generación de un archivo  
XML**

<b>DESCRIPCIÓN DEL PROGRAMA</b>	<b>2</b>
<b>Análisis de los requisitos del programa</b>	<b>2</b>
<b>Modularización del programa</b>	<b>5</b>
Lista de módulos	5
Solución de los requisitos de cada módulo	6
1. Módulo data.h	6
2. Módulo api	6
3. Módulo adapte	7
4. Módulo parser	8
5. Módulo menu	8
6. Módulo principal	8
<b>COMPILACIÓN Y EJECUCIÓN</b>	<b>9</b>
<b>Compilación sin errores</b>	<b>9</b>
<b>Ejecución</b>	<b>9</b>
1. Datos de los adaptadores de red:	9
2. Carga de datos, menú principal y selección del adaptador de red.	10
3. Imposibilidad de realizar la actualización sin haber escogido un adaptador previamente.	11
4. Obtención de los datos y creación del archivo XML.	11
5. Lectura del archivo XML en bruto.	12
6. Validación del archivo XML.	12
<b>ARCHIVOS ADJUNTOS</b>	<b>13</b>
<b>ANEXOS</b>	<b>14</b>
<b>I. Sobre el archivo XML</b>	<b>14</b>
<b>RECURSOS</b>	<b>14</b>

# AA4. Estableciendo las bases de la integración de aplicaciones mediante la generación de un archivo XML

## Descripción del programa

### Análisis de los requisitos del programa

Iniciamos, como en los productos anteriores, realizando la identificación de requisitos para proseguir, a continuación, con los pasos de clasificación de los módulos, el análisis descendente del problema y el diseño e implementación de las funciones necesarias para cada módulo. Para este primer paso, ordenaremos los requisitos en forma de árbol, con lo que facilitaremos la identificación de los módulos principales y los requisitos internos de cada uno, así podremos encontrar más rápidamente las deferentes funciones que deberá implementar cada uno.

Tras la lectura del enunciado de la actividad, podemos identificar los siguientes requisitos para el programa:

1. **Carga de datos iniciales:** El programa, inicialmente, debe obtener una lista de ellos adaptadores de red para que el usuario pueda escoger de qué adaptador quiere obtener la información.
  - a. El programa debe lanzar un comando CMD netsh para obtener la lista de los adaptadores y su número de índice. Trabajar con índices es más fiable porque evita errores por posibles caracteres especiales.
  - b. Los datos quedarán cargados en el programa.
2. **Selección de un adaptador de red:**
  - a. El programa debe disponer de los datos necesarios:
    - i. En este caso, he creído que lo mejor es que solo muestre los adaptadores activos para no realizar la prueba sobre adaptadores no conectados o sobre el *Loopback*.
    - ii. Para facilitar el proceso, en lugar de capturar el nombre del adaptador, trabajaremos a través del índice del adaptador.
  - b. La selección del adaptador se realizará mediante un menú contextual donde el usuario escogerá mediante enteros para simplificar la selección y evitar posibles errores.
  - c. Una vez escogido, debemos ser capaces de poder recuperar el índice del adaptador en cualquier momento.
3. **Obtenemos la información necesaria:**
  - a. El programa no debe iniciar este paso a no ser que haya sido escogido el adaptador de red.

- b. Deben obtenerse dos grupos de datos diferentes:
    - i. Mediante un comando *netsh* se obtendrán las diferentes direcciones del adaptador: IP, máscara de subred, puerta de enlace y servidor DNS.
    - ii. Mediante un comando *tracert* se obtendrán el número de saltos, las direcciones de cada servidor y el tiempo medio de respuesta del servidor.
  - c. Los datos deben poderse recuperar fácilmente en cualquier momento tanto para mostrarlos por pantalla como para generar el documento XML.
- 4. Generamos el documento XML:**
- a. El documento debe estar bien foarmateado y ser válido.
    - i. Tiene que incluir una cabecera con el formato XML y el enlace al archivo XSD.
    - ii. Cada elemento debe estar precedido por su etiqueta de apertura y finalizar con su etiqueta de cierre.
    - iii. EL documento debe de estar correctamente tabulado.
    - iv. Hay que cerrar el documento con las etiquetas necesarias una vez se hayan acabado de cargar los datos.

Una vez analizados los requisitos principales que se extraen del enunciado del producto 4, realizamos un segundo análisis en profundidad para incluir los requisitos que se extienden de los principales y aquellos que puedan generarse para optimizar el programa dentro de lo que haya podido generar:

- 5. La interfaz del programa se trata de un menú contextual por línea de comandos.**
- 6. Todas las estructuras y constantes se encuentran aisladas en un archivo de cabecera independiente:** De este modo se facilita su manejo.
- 7. El programa usará estructuras de datos TAD y tablas:** De este modo simplifica el manejo de datos posterior a su obtención y toda la información se encuentra ordenada. Además, todas estas estructuras deberán poderse limpiar o liberar una vez ya no necesiten usarse.
  - a. Creación de las estructuras:
    - i. Estructura TAD para almacenar los datos de un adaptador de red: Nombre, número de índice IP, máscara, puerta de enlace, servidor DNS e información del test de velocidad (saltos, tiempo medio de respuesta y dirección de cada servidor intermedio).
    - ii. Tabla de estructuras TAD de adaptador.
    - iii. Estructura TAD para almacenar la información de cada salto de servidor DNS. Esta estructura será un campo de la estructura principal de adaptador.
  - b. Inicialización de las estructuras.
  - c. Funciones necesarias para:
    - i. Añadir elementos a las estructuras.
    - ii. Búsqueda de un adaptador de red en concreto por su índice.
    - iii. Mostrar información de las estructuras.
  - d. Eliminación de las estructuras.

8. **El programa usa una API interna:** Esta gestionará la comunicación interna entre el módulo del menú y el módulo de operaciones, la carga de los datos iniciales y la inicialización de estructuras. Así, este punto se relaciona con el requisito de la carga inicial de datos. Además, simplifica el módulo principal *main* que solo deberá comunicarse con este. El módulo API gestionará la comunicación entre los diferentes módulos de funciones y el módulo menú y principal.
9. **Gestión del retorno en el test de velocidad:** En la traza a una dirección DNS podemos obtener como resultado de un salto *tiempo de espera agotado*, que marca con el carácter '\*' los tiempos de respuesta. Esto no significa que el salto no sea accesible, sino que ese servidor en específico bloquea el paquete de datos, normalmente por el *firewall*. Ignorar esta respuesta no es correcto porque enturbiaría el cálculo del tiempo de respuesta medio. Por lo tanto, se realiza una gestión de estos casos: El comando *tracert* se lanza con la opción *-w 1000*, con ello, se limita el tiempo máximo de respuesta a 1000ms; así, en caso de obtener un retorno sin respuesta, asignamos el tiempo por defecto a ese retorno. **Esto complementa el requisito 3.b.ii.**
10. **Mostrar el documento generado al usuario.**
11. **Gestión de la memoria dinámica:** Al usar apuntadores, deberemos asegurarnos de gestionar el acceso a memoria de forma eficiente, asignado y liberando memoria según sea necesario.
  - a. Este requisito genera necesidades sobre el resto de los puntos.
12. **Mantener el módulo principal lo más limpio posible.**

### Análisis descendente

Mediante la obtención de los requisitos, definimos el problema general. De este modo, procedemos a un análisis descendente del algoritmo guiándonos por el árbol de requisitos. Durante el proceso, subdividimos el problema en subproblemas menores que se pueden ir solucionando mediante subdivisiones de estos hasta obtener subproblemas lo suficientemente sencillos como para solucionar con una función única. Por definición, buscamos funciones independientes capaces de solventar una única tarea que iremos modularizando en funciones que las agrupan para solucionar los problemas de forma inversa.

El programa debe proporcionar las siguientes soluciones:

- **Problema general:** Generar un documento XML válido y bien formateado con toda la información de un adaptador de red escogido por un usuario.
- **Subproblema:** Realizar la carga de datos inicial, obtener la lista de adaptadores de red activos del equipo.
- **Subproblema:** Obtener la información de la configuración de direcciones de un adaptador escogido por el usuario.
- **Subproblema:** Realizar un test de trazabilidad sobre el servidor DNS del adaptador escogido por el usuario y almacenar todos los datos.
- **Subproblema:** Parsear toda la información en bruto a un documento XML.
- **Subproblema:** Gestión de memoria dinámica.

## Modularización del programa

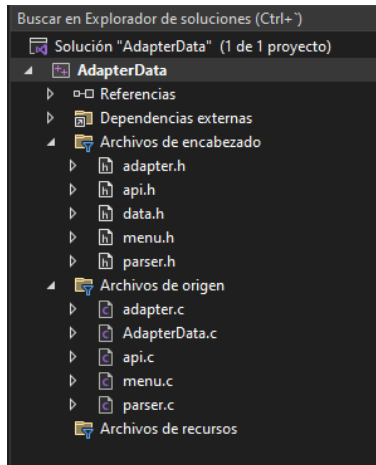
Mediante el análisis de los requisitos y la estructuración de los problemas estilo *divide & conquer*, procedemos a la identificación de los distintos módulos: Hay que crear un módulo con los archivos de cabecera, *data.h* que contendrá las constantes y estructuras de los programas. Los prototipos de todas las funciones de cada módulo se generan en un archivo de cabecera *nombreModulo.h* que se relaciona con un archivo *nombreModulo.c* donde se implementen todas las funciones.

Además, hay que enlazar las librerías de cabecera siempre que se vayan a usar en cualquier otro módulo, para ello usamos la sintaxis `#include "modulo.h"` que indica al compilador donde ir a buscar las cabeceras.

## Lista de módulos

Mediante el análisis de requisitos hallamos las funciones principales que debemos implementar y, de esta forma, los módulos del programa, el primer paso en el análisis descendente del problema. A través de la profundización en los subrequisitos y esta primera asociación, podremos ir encontrando todas las funciones que deberá implementar cada módulo que deberán ir solucionando los problemas particulares. La unión de los módulos deberá solucionar el problema principal.

1. **Módulo api** → Soluciona: Realizar la carga de datos iniciales, inicializar las estructuras y manejar la comunicación entre los módulos.
2. **Módulo adapter** → Soluciona Obtener la información de la configuración de un adaptador de red escogido por el usuario, realizar el test de velocidad sobre este y almacenar los datos.
3. **Módulo parser** → Soluciona: Generar un documento XML bien formateado y válido.
4. **Módulo menú** → Soluciona: La interfaz del programa se trata de un menú contextual por línea de comandos.
5. **Módulos data.h** → Extra: Archivo de cabeceras que aísla la declaración de las constantes y de los tipos de estructuras. No es necesario crear un .c.
6. **Módulo principal con el main: AdapterData** → Módulo para inicializar y controlar el flujo del programa y el valor de retorno. Este módulo usa un driver de control de flujo y gestiona los posibles errores mediante la actualización de un valor de retorno.



## Solución de los requisitos de cada módulo

En general, en todos los módulos he intentado trabajar siguiendo las medidas de seguridad necesarias para **gestionar la integridad de los datos**; así, aunque no se describa en cada punto en particular, se han usado bloques condicionales para controlar que no haya punteros NULL, se han usado funciones de librerías estándar más seguras, por ejemplo, *strncpy()* en lugar de *strcpy()*, o, en la declaración de constantes, se ha usado el bloque *#ifndef* para evitar repeticiones en la definición de estas y, en la creación de caberes, la cláusula *#pragma once* para asegurarnos de que una librería solo se carga una vez en la compilación. Igualmente, la apertura de archivos externos se realiza mediante la función *fopen\_s* que, según Microsoft, es una función más segura. También, como he comentado, he intentado mejorar la gestión de errores y, no solo mostrar por pantalla un mensaje de error, sino **actualizar el posible valor de retorno final del programa** llevando la gestión de una variable de retorno.

En cuanto a las constantes, su uso intenta facilitar el cambio de cualquier valor literal en el programa sin tener que revisar todo el código, línea a línea. Entre otras se indican: Ruta del archivo XML, tamaños máximos de las cadenas de caracteres, tamaños máximos de tabla, caracteres o strings de control y opciones de apertura de los archivos.

### 1. Módulo data.h

Para facilitar el trabajo con los datos que vamos obteniendo durante la ejecución del programa, tras su lectura son almacenados en diferentes tipos de estructuras.

**Lista de estructuras:** `typedef struct tJump; typedef struct tAdapter; typedef struct tAdaptersTable;`

### 2. Módulo api

Gestiona la inicialización de las diferentes estructuras asignando la memoria dinámica en caso de ser necesario y valores por defecto a los campos de las estructuras para poder gestionar

errores y, cuando sea necesario, realizar de forma correcta los cálculos para la asignación de los valores reales. Por otro lado, va realizando la delegación necesaria a funciones de los módulos *adapter*, *parser* y *menu* para la carga de datos, la creación o lectura del archivo XML y la interacción con el menú contextual, es decir, obtiene los datos necesarios mediante las funcionalidades de las funciones de los distintos módulos y los envía a donde sean requeridos.

**Lista de funciones:** `void loadData(adaptersTable* adptrsTbl, int* retVal); void initAdapter(adapter* adptr); void initAdaptersTable(adaptersTable* table); void parseAdptrInfo(const adapter adptr, int* retVal); void callMenu(adaptersTable* adptrsTbl, bool* isSalir, int* adptrInd, int* retVal); void getAdaptersData(adaptersTable* adptrsTbl, const int adptrInd, int* retVal); void closeData(int* retVal); void readData(int* retVal); void printAdaptersName(const adaptersTable table); bool adaptersTableIsFull(const adaptersTable table); void freeAdapter(adapter* adptr); int clearTable(adaptersTable* table, const int start, const int end);`

### 3. Módulo *adapter*

Este módulo se encarga de la obtención y la gestión de datos de configuración de los adaptadores de red y de la realización del test de velocidad. Es el módulo más grande y complejo, por lo tanto, he intentado guiarme a través de la creación de funciones principales y, a continuación, de todas las funciones auxiliares necesarias para implementar las principales. Como en todos los casos, gestiona la memoria dinámica, el control y manejo de errores y la eliminación de estructuras que ya no son necesarias.

Me gustaría destacar los siguientes aspectos:

- La obtención de datos se realiza mediante pipes a un puntero FILE de los comandos, así no se requiere la creación de archivos ni el uso de memoria extra.
- Se ha minimizado el uso de palabras clave en comparación con los productos anteriores, se usa control de información mediante líneas y caracteres sin contextualización a los diferentes idiomas posibles. La única palabra clave usada aparece en todos los casos en inglés, con lo que no presenta problema de posible uso de caracteres especiales.
- En cuanto a la gestión del retorno tipo *tiempo de espera agotado* en los tests de velocidad, no se pueden ignorar ya que sería un problema en el cálculo del tiempo de respuesta medio. La solución que he ideado es, por un lado, lanzar el comando *tracert* con la opción *-w 1000*, así me aseguro de que el tiempo máximo de espera a la respuesta sea de 1000 ms; por otro, capturar las líneas que devuelven el carácter '\*' y en este caso, asignar 1000 al valor de tiempo de respuesta. De este modo puedo realizar un cálculo relativo del tiempo medio de respuesta.
- Se ha intentado mejorar la gestión de errores en comparación a los productos anteriores.
- Se ha estructurado el módulo con las funciones principales en primer lugar seguidas de las funciones auxiliares.

**Funciones principales:** `void loadAdptrsInfo(adaptersTable* table, int* retVal); void getIpInfo(adaptersTable* table, const int adptrInd, int* retVal); void getDnsTest(adaptersTable* table, const int adptrInd, int* retVal);`



**Funciones auxiliares:** `void addAdaptersNameAndIndex(adaptersTable* table, const adapter adptr, int* retVal); int searchAdapter(const adaptersTable table, const int indToFind, const int start, const int end); void printAdapterInfo(const adaptersTable table, const int indPos);`

#### 4. Módulo parser

Gestiona la creación, carga de datos, cierre y lectura del archivo XML con la información del adaptador de red.

- Usa constantes para facilitar los posibles cambios futuros en el código.
- Usa funciones de apertura segura del archivo *fopen\_s()*.
- Gestiona la memoria de forma dinámica.

**Lista de funciones:** `void createXmlDoc(int* retVal); void fillXmlDoc(const adapter adptr, int* retVal); int closeXmlDoc(); int rawReadxmlDoc();`

#### 5. Módulo menu

Para poder implementar los menús, necesitamos comenzar por ser capaces de obtener la lectura de un entero introducido por un usuario. Esta función también incluye el manejo de errores si el usuario entra un entero fuera de las opciones permitidas.

A continuación, implementamos la llamada al menú principal, para ello usaremos un bloque Switch Case que lanzará un *driver* en concreto según la opción escogida por el usuario. Las siguientes funciones implementan los diferentes drivers que delegan en los distintos módulos. Todos tienen nombres que facilitan la lectura y comprensión del flujo del programa. Además, se usa un *driver* de menú y otro de submenú que implementa los bucles de los menús, así simplificamos aún más el código

**Lista de funciones:** `void mainMenu(adaptersTable* adptsTbl, bool* isSalir, int* adptrPos, int* retVal); int readInt(int limit); void driverEscogerAdaptador(const adaptersTable adptsTbl, int* adptrPos, int* retVal); void driverCargarDatosAdaptador(adaptersTable* adptsTbl, const int adptrInd, int* retVal); int driverLeerArchivoXML();`

#### 6. Módulo principal

Como en los productos anteriores, he intentado mantener el módulo con la función *main* lo más limpia posible, con la lectura más sencilla para ver cómo funciona el flujo del programa. Este módulo, implementa una función *driver* que se encarga de realizar las llamadas secuenciales a las distintas funciones principales, gestionar el valor de retorno y cerrar y limpiar los datos.

**Lista de funciones:** `int mainDriver();`

# Compilación y ejecución

## Compilación sin errores

```

Salida
Mostrar salida de: Compilación
Operación Recompilar iniciada...
1>----- Operación Recompilar todo iniciada: proyecto: AdapterData, configuración: Debug Win32 -----
1>adapter.c
1>AdapterData.c
1>api.c
1>menu.c
1>parser.c
1>Generando código...
1>AdapterData.vcxproj -> C:\Users\D. Ryalbran\Desktop\p4\AdapterData\Debug\AdapterData.exe
===== Recompilar todo: 1 correcto, 0 con errores, 0 omitido =====
===== 00:02,004 Transcurrido =====
  
```

## Ejecución

### 1. Datos de los adaptadores de red:

```

C:\Users\D. Ryalbran>netsh interface ipv4 show interfaces

Índ    Mét      MTU      Estado      Nombre
-----
1      75  4294967295 connected Loopback Pseudo-Interface 1
9      55      1500 connected Wi-Fi
8      65      1500 disconnected Conexión de red Bluetooth
4      25      1500 disconnected Conexión de área local* 1
15     25      1500 disconnected Conexión de área local* 2
20     25      1500 connected VirtualBox Host-Only Network
41     15      1500 connected vEthernet (WSL)

C:\Users\D. Ryalbran>
  
```

```

C:\Users\D. Ryalbran>netsh interface ipv4 show config 9

Configuración para la interfaz "Wi-Fi"
DHCP habilitado: No
Dirección IP: 192.168.1.22
Prefijo de subred: 192.168.1.0/24 (máscara 255.255.255.0)
Puerta de enlace predeterminada: 192.168.1.1
Métrica de puerta de enlace: 0
Métrica de interfaz: 55
Servidores DNS configurados estáticamente: 9.9.9.9
Registrar con el sufijo: Solo el principal
Servidores WINS configurados estáticamente: ninguno
  
```

## 2. Carga de datos, menú principal y selección del adaptador de red.

```
C:\Users\D. Ryalbran\Desktop\p4\AdapterData_programa\AdapterData.exe

=====
Bienvenido al programa XML Adapter Info.
=====

-----
1. Escoger adaptador de red.
2. Generar documento XML con los datos del adaptador.
3. Mostrar archivo XML generado.
0. Salir
Escoja 1, 2, 3 o 0:
-----
1

-----
Lista de adaptadores de red de este equipo:
-----
Num      Nombre Adaptador de Red
1.        Wi-Fi
2.        VirtualBox Host-Only Network
3.        vEthernet (WSL)
-----
Por favor, indique el numero de adaptador del que desea obtener informacion.
Escoja un entero entre 1 y 3:
1
-----
Informacion del adaptador escogido:
Nombre: Wi-Fi
Indice de adaptador: 9

-----
1. Escoger adaptador de red.
2. Generar documento XML con los datos del adaptador.
3. Mostrar archivo XML generado.
0. Salir
Escoja 1, 2, 3 o 0:
```

### 3. Imposibilidad de realizar la actualización sin haber escogido un adaptador previamente.

```

C:\Users\D. Ryalbran\Desktop\p4\AdapterData_programa\AdapterData.exe
=====
Bienvenido al programa XML Adapter Info.
=====
1. Escoger adaptador de red.
2. Generar documento XML con los datos del adaptador.
3. Mostrar archivo XML generado.
0. Salir
Escoja 1, 2, 3 o 0:
2
=====
Por favor, seleccione un adaptador de red antes de continuar.
=====
1. Escoger adaptador de red.
2. Generar documento XML con los datos del adaptador.
3. Mostrar archivo XML generado.
0. Salir
Escoja 1, 2, 3 o 0:

```

### 4. Obtención de los datos y creación del archivo XML.

```

C:\Users\D. Ryalbran\Desktop\p4\AdapterData_programa\AdapterData.exe
=====
1. Escoger adaptador de red.
2. Generar documento XML con los datos del adaptador.
3. Mostrar archivo XML generado.
0. Salir
Escoja 1, 2, 3 o 0:
2
=====
Realizando el test de velocidad a los servidores DNS:
(Las acciones pueden demorarse un poco)
=====
Calculo de la traza al servidor DNS 9.9.9.9 en proceso:
*****
=====
Datos del adaptador de red seleccionado:
Nombre: Wi-Fi
Direccion IP: 192.168.1.22
Mascara de subred: 255.255.255.0
Puerta de enlace: 192.168.1.1
DNS: 9.9.9.9
Tiempo medio de respuesta: 235.00
Saltos totales: 10
Numero de salto: 1, Direccion del servidor: 192.168.1.1
Numero de salto: 2, Direccion del servidor: 10.14.32.1
Numero de salto: 3, Direccion del servidor: 10.10.30.5
Numero de salto: 4, Direccion del servidor: 10.10.30.1
Numero de salto: 5, Direccion del servidor: 172.16.30.5
Numero de salto: 6, Direccion del servidor: 10.10.20.73
Numero de salto: 7, Direccion del servidor: 10.97.30.241
Numero de salto: 8, Direccion del servidor: Servidor bloqueado!
Numero de salto: 9, Direccion del servidor: pch42.baja.espanix.net [193.149.1.120]
Numero de salto: 10, Direccion del servidor: dns9.quad9.net [9.9.9.9]
=====
1. Escoger adaptador de red.
2. Generar documento XML con los datos del adaptador.
3. Mostrar archivo XML generado.
0. Salir

```

## 5. Lectura del archivo XML en bruto.





```

C:\Users\D. Ryalbran\Desktop\p4\AdapterData_programa\AdapterData.exe
0. Salir
Escoja 1, 2, 3 o 0:
-----
3
<?xml version='1.0' encoding='UTF-8'?>

<adapter_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./adapter_info.xsd">
  <name>Wi-Fi</name>
  <ip>192.168.1.22</ip>
  <mask>255.255.255.0</mask>
  <gateway>192.168.1.1</gateway>
  <dns_server>9.9.9.9</dns_server>
  <jump>Informacion de los saltos:
    <jump_number>1</jump_number>
    <ip>192.168.1.1 </ip>
    <jump_number>2</jump_number>
    <ip>10.14.32.1 </ip>
    <jump_number>3</jump_number>
    <ip>10.10.30.5 </ip>
    <jump_number>4</jump_number>
    <ip>10.10.30.1 </ip>
    <jump_number>5</jump_number>
    <ip>172.16.30.5 </ip>
    <jump_number>6</jump_number>
    <ip>10.10.20.73 </ip>
    <jump_number>7</jump_number>
    <ip>10.97.30.241 </ip>
    <jump_number>8</jump_number>
    <ip>Servidor bloqueado!</ip>
    <jump_number>9</jump_number>
    <ip>pch42.baja.espanix.net [193.149.1.120] </ip>
    <jump_number>10</jump_number>
    <ip>dns9.quad9.net [9.9.9.9] </ip>
  </jump>
</adapter_info>
-----
1. Escoger adaptador de red

```

## 6. Validación del archivo XML.

Nombre	Fecha de modificación	Tipo	Tamaño
 adapter_info.xsd	17/12/2022 22:48	XML Schema File	1 KB
 AdapterData	21/12/2022 9:34	Aplicación	59 KB
 AdapterData.pdb	21/12/2022 9:34	Program Debug D...	1.020 KB
 adpater_info	21/12/2022 9:40	Documento XML	1 KB

```
<?xml version='1.0' encoding='UTF-8'?>

<adapter_info xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="./adapter_info.xsd">
  <name>Wi-Fi</name>
  <ip>192.168.1.22</ip>
  <mask>255.255.255.0</mask>
  <gateway>192.168.1.1</gateway>
  <dns_server>9.9.9.9</dns_server>
  <jump>Informacion de los saltos:
    <jump_number>1</jump_number>
    <ip>192.168.1.1 </ip>
    <jump_number>2</jump_number>
    <ip>10.14.32.1 </ip>
    <jump_number>3</jump_number>
    <ip>10.10.30.5 </ip>
    <jump_number>4</jump_number>
    <ip>10.10.30.1 </ip>
    <jump_number>5</jump_number>
    <ip>172.16.30.5 </ip>
    <jump_number>6</jump_number>
    <ip>10.10.20.73 </ip>
    <jump_number>7</jump_number>
    <ip>10.97.30.241 </ip>
    <jump_number>8</jump_number>
  </jump>
</adapter_info>
```

www.w3schools.com dice  
No errors found  
Aceptar

Try to syntax-check your own XML :

```
<jump_number>7</jump_number>
<ip>10.97.30.241 </ip>
<jump_number>8</jump_number>
<ip>Servidor bloqueado!</ip>
<jump_number>9</jump_number>
<ip>pch42.baja.espanix.net [193.149.1.120] </ip>
<jump_number>10</jump_number>
<ip>dns9.quad9.net [9.9.9.9] </ip>
</jump>
</adapter_info>
```

## Archivos adjuntos

Junto con los archivos requeridos por la actividad, he incluido dos archivos más que pienso pueden resultar útiles:

1. **Directorio con el archivo ejecutable por separado**, está en una carpeta a parte para que sea más sencilla su ejecución.
2. **Documentación de todas las estructuras y funciones del programa**: Además de incluirla en los comentarios, he pensado que resultaría útil extraerla a un archivo externo.

# Anexos

## I. Sobre el archivo XML

En el caso de que, en una única ejecución del programa, se acceda a la configuración de distintos adaptadores, estos se irán agregando uno tras otro en el documento final. En cambio, una nueva ejecución del programa generará un archivo XML nuevo borrando la información anterior.

El programa necesita que, en la carpeta raíz, este presente el archivo *adapter\_info.xsd* donde se implementa el esquema XML.

## Recursos

C library function - *fgets()*. (s. f.). [https://www.tutorialspoint.com/c\\_standard\\_library/c\\_function\\_fgets.htm](https://www.tutorialspoint.com/c_standard_library/c_function_fgets.htm)

MSFT, T. (2022, 22 octubre). *sprintf\_s, \_sprintf\_s\_l, swprintf\_s, \_swprintf\_s\_l*. Microsoft Learn.

<https://learn.microsoft.com/es-es/cpp/c-runtime-library/reference/sprintf-s-sprintf-s-l-swprintf-s-swprintf-s-l?view=msvc-170>

MSFT, T. (2022b, octubre 27). *printf\_s, \_printf\_s\_l, wprintf\_s, \_wprintf\_s\_l*. Microsoft Learn.

<https://learn.microsoft.com/es-es/cpp/c-runtime-library/reference/printf-s-printf-s-l-wprintf-s-wprintf-s-l?view=msvc-170>

C library function - *strncpy()*. (s. f.). [https://www.tutorialspoint.com/c\\_standard\\_library/c\\_function\\_strncpy.htm](https://www.tutorialspoint.com/c_standard_library/c_function_strncpy.htm)

MSFT, T. (2022c, diciembre 2). *\_popen, \_wopen*. Microsoft Learn. <https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/popen-wopen?view=msvc-170>

MSFT, T. (2022d, diciembre 2). *strncpy\_s, \_strncpy\_s\_l, wcsncpy\_s, \_wcsncpy\_s\_l, \_mbstrncpy\_s, \_mbstrncpy\_s\_l*.

Microsoft Learn. <https://learn.microsoft.com/es-es/cpp/c-runtime-library/reference/strncpy-s-strncpy-s-l-wcsncpy-s-wcsncpy-s-l-mbstrncpy-s-mbstrncpy-s-l?view=msvc-170>

MSFT, T. (2022b, octubre 26). *fprintf\_s, \_fprintf\_s\_l, fwprintf\_s, \_fwprintf\_s\_l*. Microsoft Learn.

<https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/fprintf-s-fprintf-s-l-fwprintf-s-fwprintf-s-l?view=msvc-170>

MSFT, T. (2022d, diciembre 2). *fopen\_s, \_wfopen\_s*. Microsoft Learn. <https://learn.microsoft.com/es-es/cpp/c-runtime-library/reference/fopen-s-wfopen-s?view=msvc-170>