

ACTIVIDAD 2. XML, DTD Y XSD

LENGUAJE DE MARCAS



VERÓNICA BONIS MARTÍN

MARIA CARMEN CORREA HERAS

ÁNGEL SÁNCHEZ-SIERRA CRUZ

JOSÉ MARÍA TENREIRO EIRANOVA

JUAN RAMON VARÓ NÚÑEZ

REPOSITORIO GITHUB

[HTTPS://GITHUB.COM/ITDDAW/ACTIVIDAD2LDM.GIT](https://github.com/ITDDAW/ACTIVIDAD2LDM.GIT)

Vamos a elaborar una estructura de base de datos en XML que permita almacenar los datos de una biblioteca en la red. Para ello se pide que elabores un DTD que permita validar los documentos XML con las siguientes características:

- Existen tres tipos de documentos almacenados en la biblioteca: libros, revistas y periódicos. Todos los documentos están identificados por el atributo `Id`.
 - Para los libros este atributo empieza con la letra "L" seguido de 4 dígitos identificativos.
 - Para los periódicos este atributo empieza por la letra "P" seguido de los 4 dígitos identificativos.
 - En el caso de las revistas empieza por la letra "R".
- Los libros a su vez son clasificados en novela, infantil o didáctico. Cada libro contiene un atributo identificativo de su clase denominado `tipo_clase`. Dentro de cada libro se tiene un título, varios capítulos con el título en su interior, un índice y una sinopsis. Tanto en libro como en capítulo existe un atributo que contiene el número de páginas del libro o del capítulo, según corresponda. Cada capítulo contiene un elemento denominado `contenido`, en el que se tiene un atributo con el enlace a la información.
- Las revistas a su vez son clasificadas en: informática, corazón, coches, investigación y otras. Cada revista tiene el atributo `tipo_clase` identificativo de la clase a la que pertenece. Dentro de cada revista tenemos el título, el número de la revista, un índice de contenido y las secciones. En cada sección y en la revista se tiene un atributo que contiene el número de páginas. Además, en cada sección se tiene la parte denominada `contenido`, en la que se tiene un atributo con un enlace a la información. De igual forma que en los anteriores, la última etiqueta del árbol debe ser el `contenido` con un atributo que referencia al contenido.
- Los periódicos se clasifican en nacionales e internacionales. Cada periódico contiene el atributo `tipo_clase` identificativo de la clase a la que pertenece y un atributo que incluye la fecha de publicación.
- Dentro de los periódicos tenemos secciones y un índice. Cada sección debe contener un atributo identificativo del tipo de sección, que puede ser: económica, opinión, deportes, nacional o internacional. Las secciones se dividen en artículos, en donde se define en un atributo el autor. Finalmente, el contenido será el último elemento del árbol, que necesita un atributo que referencia a la información.

Requerimiento 1

- 1) Crear un XML con el modelo de datos indicado en el enunciado.
- 2) Elaborar un DTD que permita validar el documento XML.
- 3) Validar el documento con alguna aplicación externa.

IMPORTANTE: Cada alumno propondrá una solución de XML y DTD. Una vez todos los alumnos hayan puesto todas sus soluciones, se cotejará con el resto para poner la solución final.

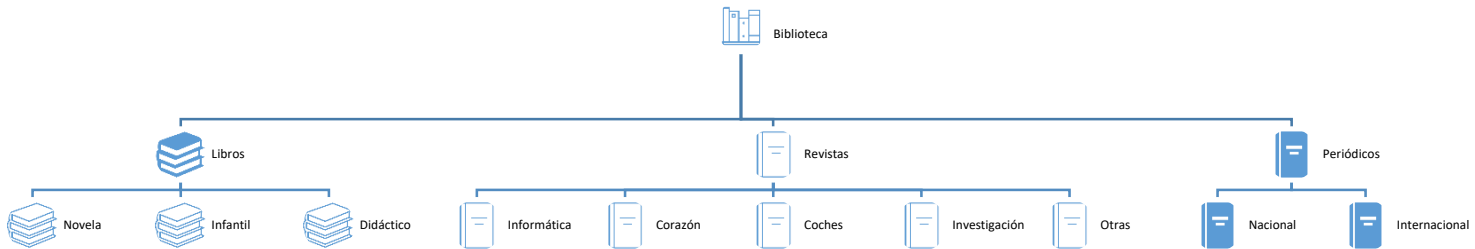
CONSIDERACIONES PREVIAS

Para completar la tarea, cada persona trabaja por separado su versión de XML. Se hacen unas consideraciones previas:

- Trabajar en Visual Studio Code. Se recomienda instalar el plugin [XML Language Support by Red Hat](#) que nos facilitará la tarea de dar formato al archivo XML, corrección de errores, asignar sangrías automáticamente a los nodos, y que nos parece muy buena opción para trabajar.
- Probamos también otro editor específico: EDITIX [Professional XML Editor 2021 - EditiX](#), que ofrece algunas ventajas y que tiene una versión gratuita bajo algunas condiciones
- Trabajar de nuevo con GIT, subiendo cada comentario y documentos a la carpeta compartida en el repositorio <https://github.com/ITDDAW/Actividad2LdM.git>

Una vez preparado el entorno de trabajo y el método, comenzamos a analizar el problema que nos plantea el ejercicio. Para ello nos parece interesante seguir este flujo:

- a) Crear una “estructura de árbol” en la que identifiquemos claramente la estructura jerárquica de los elementos que se incluirán en el XML (nodo raíz, subnodos, atributos...) que nos permita ir definiendo mejor las etiquetas.
- b) Definir el nodo raíz y los subnodos mediante etiquetas.
- c) Ir rellenando los atributos requeridos en cada elemento.



FUNDAMENTACIÓN

XML no es un lenguaje sino un metalenguaje ya que permite crear lenguajes.

XML es un estándar desarrollado por el W3C [Estándares - W3C España](#), el cual desarrolla numerosas tecnologías que están basadas en XML.

XML proporciona una serie de reglas para que cualquiera pueda definir su propio conjunto de etiquetas y atributos, así como las relaciones entre las etiquetas. A esto se le llama la gramática del lenguaje.

Existen diferentes formas de definir la gramática del lenguaje basado en XML, como DTD o XML Schema, que serán las utilizadas en este ejercicio.

DTD es una descripción de la estructura y la sintaxis de un documento XML, es decir, de su gramática. DTD se ha heredado de SGML y su sintaxis no es XML. Este problema se resuelve con el uso de XML Schema.

FUNCIÓN DE UN DTD

- Describir la estructura de documentos
- Mantener la consistencia de documentos
- Permite la validación

ESTRUCTURA DE UN DTD

- Define los elementos permitidos y el contenido de dichos elementos.
- Define los atributos y valores válidos
- Define la estructura válida, el orden de los elementos.
- Declaración

<!>

- 4 tipos de declaraciones:
 - Element
 - Attlist
 - Entity
 - Notation

En primer lugar, debemos preocuparnos de que el XML **esté bien formado**, esto es:

- Debe aparecer la declaración XML lo que se llama el prólogo:

`<?xml version = "1.0" encoding = "ISO-8859-1" standalone = "yes"?>`

- Si no se incluye los valores por defecto son: version 1.0, encoding UTF-8 y standalone yes.
- El cuerpo del documento se compone de elementos con etiqueta de apertura y de cierre. La etiqueta de inicio puede tener atributos con valor entrecomillado. Y por último entre las etiquetas de inicio y cierre el elemento puede tener contenido.
- *Solo puede existir un único elemento raíz*, del que parten los demás.
- Los elementos deben seguir una estructura jerárquica. Cada elemento debe estar contenido dentro de otro, exceptuando el elemento raíz.
- Los elementos deben *estar alineados*; deben contener tabulaciones de forma que en la columna que tenemos la etiqueta de inicio de un elemento, debe estar también la de cierre.
- Los elementos no pueden estar superpuestos ni solapados.
- Los elementos *deben estar cerrados*, es decir, todos los elementos deben contener su etiqueta de inicio y de cierre.
- El nombre de los elementos debe empezar siempre con un carácter no numérico y nunca puede comenzar por XML o cualquier variación, es decir, no puede empezar por xml, Xml, XML, xML, etc.
- Hay atributos reservados:
 - El prefijo xml se reserva para las especificaciones de xml
 - Xml : lang -----> para el idioma del elemento.
 - Xml : space : default | preserved -----> procesamiento normal o conservar los espacios en blanco.
 - Xml : id -----> se emplea como identificador único de todo el documento.
- Hay que respetar las mayúsculas y minúsculas. XML es un lenguaje case sensitive (sensible a mayúsculas y minúsculas).
- Existen caracteres especiales que se deben expresar con entidades específicas.
- Los valores de los atributos deben ir entre comillas dobles o simples. Si en el valor de un atributo se quieren escribir comillas dobles o simples se deben usar estas entidades que representan caracteres especiales:

`"` `'`

- Se permiten elementos vacíos y se pueden escribir:

`<libro></libro>` ó

`<libro/>`

- Los comentarios se escriben `<!-- comentario -->`

Luego debemos comprobar que además de bien formado **es válido** (ver si cumple con las definiciones que se han incluido en el DTD y XSD).

VALIDACION MEDIANTE DTD

Para la validación se comprueba si el documento XML está hecho según el patrón definido en el DTD. Hay que distinguir entre un documento validado y un documento bien formado. El documento bien formado es simplemente el que cumple todas las reglas de XML para formar documentos. El documento validado debe, además, adaptarse al patrón que le marca la DTD.

Se podría poner el DTD insertado en el mismo documento XML, nosotros, sin embargo, lo haremos mediante un archivo externo y por tanto el contenido del DOCTYPE estará en ese archivo.

```
<!DOCTYPE saludo SYSTEM "biblioteca.dtd">
```

Empezamos a ver cómo se construye el documento DTD. Este documento indica qué etiquetas y atributos debe tener el documento XML, y cómo deben ser estas.

Las etiquetas de XML se indican mediante una etiqueta en DTD que empieza por la palabra !ELEMENT. Cuando una etiqueta tiene etiquetas hijos, la etiqueta principal no puede llevar texto (pero si atributos). Al indicar las etiquetas dependientes dentro del paréntesis debemos indicar las veces que se puede repetir cada etiqueta. De no indicar nada ésta se deberá poner una única vez.

Debemos indicar también qué atributos tiene cada elemento y sus características. Inmediatamente después de declarar un elemento debemos declarar sus atributos. Los atributos de un elemento se declaran todos dentro de la misma etiqueta <!ATTLIST.

Con la etiqueta CDATA podemos poner cualquier texto siempre que esté conforme con las reglas de escritura para XML.

Con la etiqueta ID se debe de cumplir que el atributo debe empezar por una letra, y el valor debe ser distinto en cada uno de los elementos que tengan este atributo.

VALIDACION MEDIANTE XSD

XSD (*XML Schema Definition*) es un lenguaje, también llamado simplemente XML Schema, que sirve para definir la estructura de un documento XML, permitiendo su validación.

Para validar mediante un XSD nuestro XML tenemos que hacer referencia a nuestro archivo XSD externo mediante el siguiente código:

```
<biblioteca nombre="Biblioteca Pública Madrid" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="XSDBiblioteca.xsd">
```

Actividad 2. XML, DTD y XSD

- Indicamos que también pueden aparecer elementos del namespace "http://www.w3.org/2001/XMLSchema-instance", y deben ir precedidos por el prefijo "xsi".
- Indicamos la ubicación del esquema XSD contra el que se debe validar el documento XML ("XSDBiblioteca.xsd").

Utilizaremos tipos de datos simples mediante la siguiente sintaxis: xs:string, xs:integer, xs:ID, etc.

Para indicar el tipo de un atributo, por ejemplo:

```
<xs:attribute name="lang" type="xs:string" use="required"/>
```

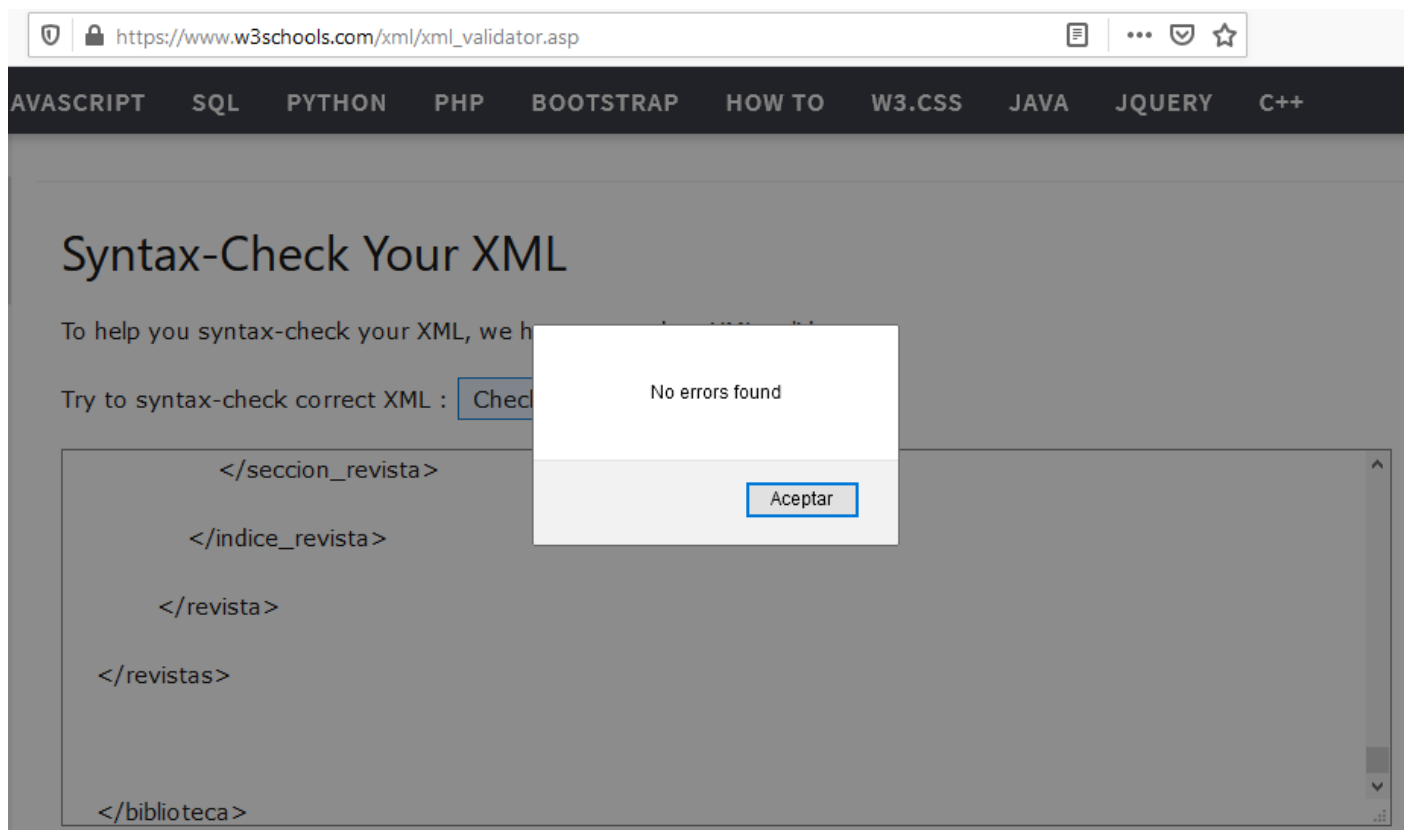
Con xs:sequence – Los elementos que contiene deben aparecer exactamente en el mismo orden en que están definidos.

Con maxOccurs="unbounded", el elemento puede aparecer un número indefinido de veces.

Se llega a la conclusión de que el orden a la hora de redactar los documentos DTD y XSD es lo más importante ya que cualquier error al final te llevará horas encontrar donde te has equivocado.

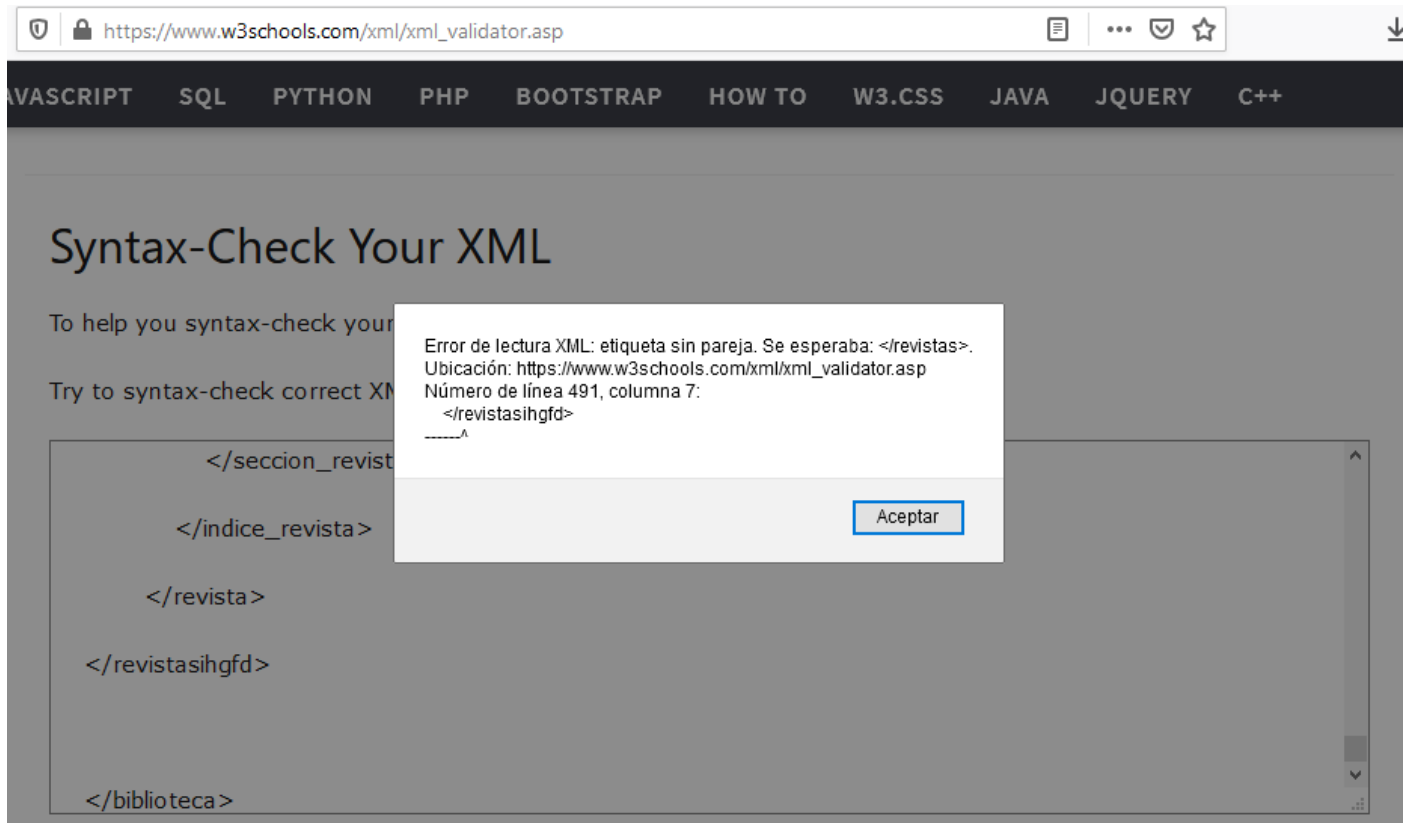
VALIDADORES EXTERNOS

Como validador externo de solamente XML utilizamos la página: www.w3schools.com



Actividad 2. XML, DTD y XSD

Como ejemplo, si en nuestro código introducimos algún error, nos daría un mensaje indicándonos donde hemos cometido el error:



Actividad 2. XML, DTD y XSD

Como validador externo de nuestro XML con archivo DTD utilizamos la página xmlvalidator.new-studio.org, en ella, introducimos nuestro archivo xml pero tenemos que ponerle el dtd interno, para ello, hacemos la prueba de ponerlo en el documento y vemos el resultado que nos ofrece:

☐ XPath ☐ Schema ☒ DTD
Charset:
Choose XML file to validate: BibliotecaDTD.xml
Due to the technical issue, embedded DTD is allowed ONLY.

XPath Explorer

Validation Result











Validation successful!

Source Code

```
view plain print 7
1. <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2.
3. <!--Esta primera línea tiene que ir sin ningún espacio ni nada anteriormente escrito, ni tan siquiera un comentario-->
4. <!--standalone=no nos indica que el documento depende
5. de un documento de validación externo-->
6.
7.
8. <!--
9. Vamos a elaborar una estructura de base de datos en XML que permita almacenar los datos de una biblioteca en la red. Para el
10. Existen tres tipos de documentos almacenados en la biblioteca: libros, revistas y periódicos. Todos los documentos están ide
11. Para los libros este atributo empieza con la letra "L" seguido de 4 dígitos identificativos.
12. Para los periódicos este atributo empieza por la letra "P" seguido de los 4 dígitos identificativos.
13. En el caso de las revistas empieza por la letra "R".
14. -->
15.
16.
17.
18. <!DOCTYPE biblioteca [
19.
20. <!ELEMENT biblioteca (libros,periodicos,revistas)>
21. <!--
22. <!--
23. <!--
24. Vamos a elaborar una estructura de base de datos en XML que permita almacenar los datos de una biblioteca en la red. Para el
25. Existen tres tipos de documentos almacenados en la biblioteca: libros, revistas y periódicos. Todos los documentos están ide
26. Para los libros este atributo empieza con la letra "L" seguido de 4 dígitos identificativos.
27. Para los periódicos este atributo empieza por la letra "P" seguido de los 4 dígitos identificativos.
28. En el caso de las revistas empieza por la letra "R".
29. Los libros a su vez son clasificados en novela, infantil o didáctico. Cada libro contiene un atributo identificativo de su c
```

Actividad 2. XML, DTD y XSD

Para validar externamente nuestro xml con el xsd creado, utilizamos también la misma página que con el anterior ejemplo, introducimos nuestros dos archivos xml y xsd y nos muestra el resultado si tenemos errores o está correctamente.

  xmlvalidator.new-studio.org      

XML Validator Online BETA

XML Validation

☐ XPath ☒ Schema ☐ DTD


Charset:

Choose XML file to validate: BibliotecaDTD.xml

XML Schema: XSDBiblioteca.xsd

XPath Explorer

Validation Result

 Validation successful!

PASOS PARA TRABAJAR EN GRUPO

La forma que ha tenido de trabajar el grupo ha sido mediante el repositorio GIT, en este repositorio se ha ido actualizando la página según cada miembro iba aportando algo a ella. Los pasos seguidos para ello en git han sido los siguientes:

- 1- Hay que decidir si se va a trabajar subiendo cambios con el perfil de Github del perfil creado (utilizar las contraseñas y nombre de usuario propias de ese perfil ITTDAW@gmail.com) o utilizar nuestro propio perfil de Github y trabajar como colaborador (Se han enviado invitaciones a todos a los correos con los que cada persona se ha registrado en Github, hay que abrirlo y darle a aceptar)
- 2- Hacer un clone del repositorio en una carpeta de nuestro equipo
Git clone <https://github.com/ITTDAW/Actividad2LdM.git>
Entrar en la carpeta que se ha creado nueva dentro e iniciar el repositorio
- 3- Ahora estaremos en la rama master por defecto de nuestro repositorio local. Si se va a realizar cambios en el programa, se recomienda primero crear una rama propia para tener dos copias por si acaso.
Git checkout -b nombrerama
- 4- Ahora ya estaremos en una nueva rama de nuestro repositorio local. Realizar los cambios que consideremos en el programa que estamos haciendo, todas las pruebas que consideremos, es el momento de probar todo.
- 5- Cuando pensamos que hemos hecho algo valido para subir al repositorio, vamos a añadir nuestros cambios en nuestro repositorio local y rama propia.
Git add .
Git commit -m "Los cambios que hemos hecho"
- 6- Ahora ya tenemos nuestros cambios en nuestro repositorio local y rama propia, podríamos verlos con git log --oneline. Para subir estos cambios al repositorio remoto, es necesario que nos cambiemos a la rama master y fusionar nuestra rama personal con la master.
Git checkout master
Git merge nombredenuestrarama
- 7- En estos momentos nuestra rama master ya puede subirse al repositorio remoto. Si lo hacemos desde nuestro perfil de github, hemos aceptado el correo de aceptación de colaboradores y en nuestro equipo ya tenemos las credenciales de nuestro perfil de github, solamente con git push origin master se deberían de subir los cambios. Si trabajamos desde el perfil de ITTDAW tendremos que loguearnos cuando lo pida el gitbash y no tener guardadas nuestras credenciales en Windows de nuestro perfil personal.
- 8- CONSEJOS: Siempre antes de empezar a trabajar, hacer git pull en la rama master de tu repositorio local para ver si algún compañero ha hecho algún cambio, después fusionar esos cambios con tu rama personal y entonces si empezar a trabajar.
- 9- Si a alguien se le complica su repositorio local, que no se preocupe, se hace otro git clone en su equipo y vuelve a empezar desde el punto 2.