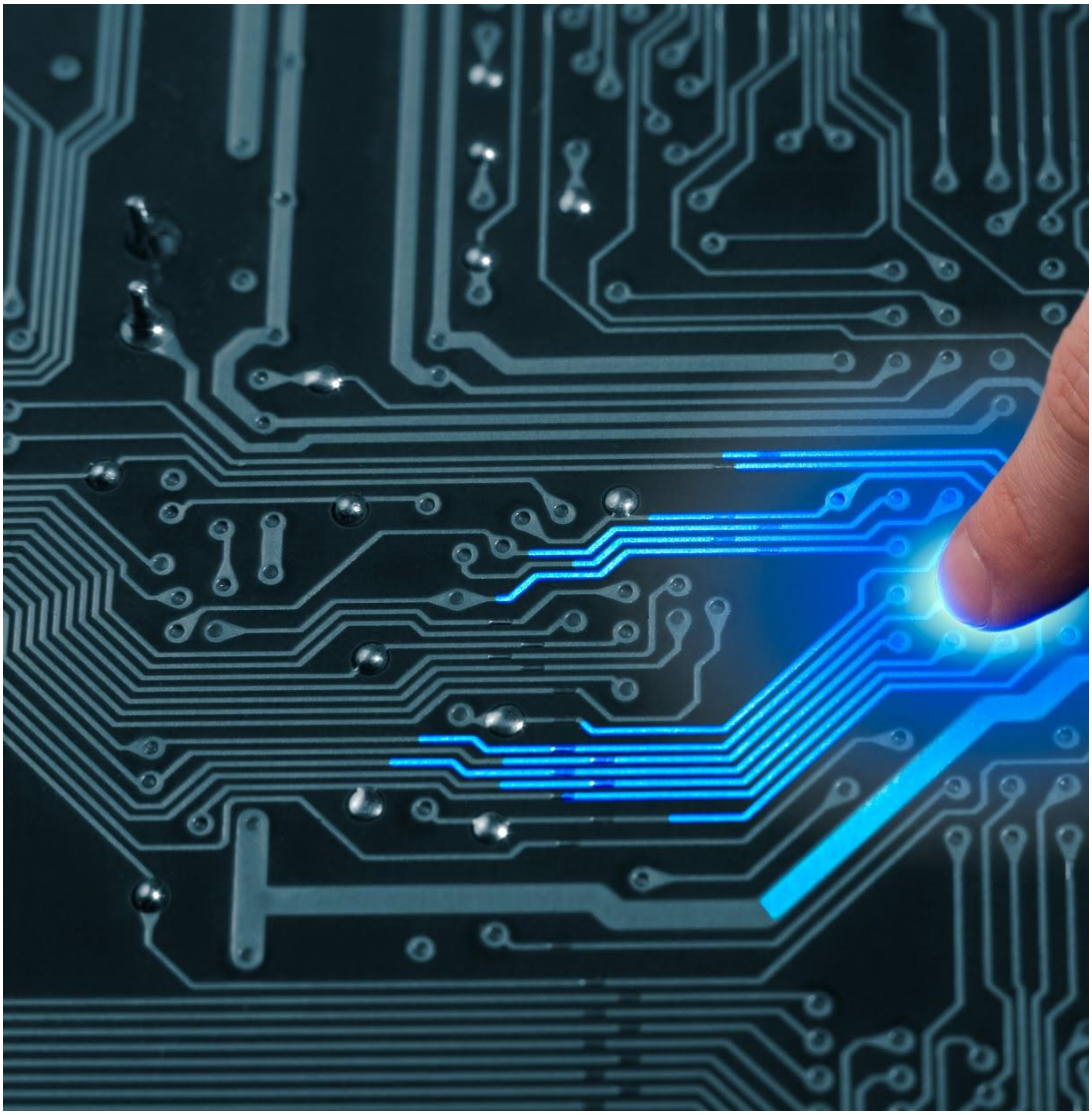
CFGS

DESARROLLO DE APLICACIONES MULTIPLATAFORMA

ACCESO A DATOS

PERSISTENCIA EN BD NATIVAS XML

PAC 1: Bases de datos XML



PAC 1: Bases de datos XML

INFORMACIÓN

Para responder a las siguientes cuestiones deberás ayudarte del material didáctico y consultar internet.

Requisitos varios que deben cumplirse en vuestros trabajos:

- En los ejercicios, si se requieren de cálculos, estos deben aparecer en la respuesta que planteéis.
- Siempre que utilicéis información de Internet para responder / resolver alguna pregunta, tenéis que citar la fuente (la página web) de dónde habéis sacado aquella información.
- Siempre que utilicéis información del libro digital para responder / resolver alguna pregunta, tenéis que citar el tema y la página de dónde habéis sacado aquella información.
- No se aceptarán respuestas sacadas de Internet utilizando la metodología de copiar y pegar. Podéis utilizar Internet para localizar información, pero el redactado de las respuestas ha de ser vuestro.
- Las respuestas a las preguntas deben estar bien argumentadas, no se admiten respuestas escuetas o monosílabas.
- Si el proyecto no compila, la puntuación máxima será un 4.
- Se valorará la presentación, ortografía y gramática de vuestro trabajo hasta con un punto de la nota final.

Entrega de la PAC:

- Se debe entregar junto con el proyecto comprimido, una memoria en el que se expliquen los pasos realizados junto con los puntos más importantes del código explicados.

Ejercicio práctico

Ejercicio 1:

- Obtén los nodos denominación y precio de todos los productos:
  /productos/produc/denominacion | /productos/produc/precio

- Obtén los nodos de los productos que sean placas base:
  /productos/produc [starts-with(denominacion,'Placa Base')]

- Obtén los nodos de los productos con precio mayor de 60 € y de la zona 20:
  /productos/produc[precio>60 and cod_zona=20]

- Obtén el número de los productos que sean memorias y de la zona 10:
  count(/productos/produc[starts-with(denominacion,'Memoria') and cod_zona=10])

- Obtén la media de precio de los micros:
  avg(/productos/produc[starts-with(denominacion,'Micro')]/precio)

- Obtén los datos de los productos cuyo stock mínimo sea mayor que su stock actual:
  /productos/produc[number(stock_minimo)>number(stock_actual)]

- Obtén el nombre de producto y el precio de aquellos cuyo stock mínimo sea mayor que su stock actual y sean de la zona 40:

  /productos/produc[number(stock_minimo)>number(stock_actual) and cod_zona=40]/denominacion |
  /productos/produc[number(stock_minimo)>number(stock_actual) and cod_zona=40]/precio

- Obtén el producto más caro:
  /productos/produc[precio=max(/productos/produc/precio)]

- Obtén el producto más barato de la zona 20:
  /productos/produc[precio=min(/productos/produc[cod_zona="20"]/precio) and cod_zona="20"]

- Obtén el producto más caro de la zona 10:
  /productos/produc[precio=max(/productos/produc[cod_zona="10"]/precio) and cod_zona="10"]

Ejercicio 2:

- Obtén por cada zona el número de productos que tiene:

```
for $produc in distinct-values(/productos/produc/cod_zona)
    let $cu:=count(/productos/produc[cod_zona=$produc])
return concat('Hay:', $cu , 'productos', 'en la zona nº',$produc)
```

- Obtén la denominación de los productos entre las etiquetas 10 si son del código 10, 20 si son de la zona 20, 30 si son de la zona 30, 40 si son de la zona 40:

```
for $prod in //produc
return
concat('<zona',$prod/cod_zona,'>',$prod/denominacion,'</zona',$prod/cod_zona,
'>')
```

- Obtén por cada zona la denominación de los productos más caros:

```
for $prod in //produc for $zon in $prod/cod_zona
    let $den:=$prod/denominacion  let $precio:=$prod/precio
return if ($precio=max(/productos/produc[cod_zona=$zon]/precio) and
/productos/produc/cod_zona=$zon)
then
< zona>{data($zon)}<preciomax>{data($precio)}</preciomax><producto>{data (
$den)}</producto></zona>
 else()
```

- Obtén la denominación de los productos contenida entre las etiquetas placa para los productos en cuya denominación aparece la palabra Placa Base, <memoria> para los que contienen la palabra memoria, <micro> para los que contienen la palabra Micro y <otros> para el resto de productos:

```
for $prod in (/productos/produc/denominacion)
    let $deno:=data(/productos/produc[contains (denominacion,
$prod)]/denominacion)
return if (contains($prod, 'Placa Base'))  then <placa>{$deno}</placa> else(
if (contains($prod, 'Memoria'))  then <memoria>{$deno}</memoria> else(
if (contains($prod, 'Micro'))  then <micro>{$deno}</micro> else()))
```

- Devuelve el código de sucursal y el número de cuentas que tiene de tipo ahorro y de tipo pensiones.

```
for $sucursal in /sucursales/sucursal
    let $numeroAhorro := count($sucursal/cuenta[@tipo = 'AHORRO'])
    let $numeroPensiones := count($sucursal/cuenta[@tipo = 'PENSIONES'])
return <Sucursal> <Codigo>{data($sucursal/@codigo)}</Codigo>
<Ahorro>{$numeroAhorro}</Ahorro>
<Pensiones>{data($numeroPensiones)}</Pensiones> </Sucursal>
```

- Devuelve por cada sucursal el código de sucursal, el director, la población, la suma del total debe y la suma de total haber de sus cuentas:

```
for $suc in distinct-values(//sucursales/sucursal/@codigo)
    let $codigo:=//sucursales/sucursal[@codigo=$suc]/@codigo
    let $director:=/sucursales/sucursal[@codigo=$suc]/director
    let $poblac:=/sucursales/sucursal[@codigo=$suc]/poblacion
    let $haber:=sum(/sucursales/sucursal[@codigo=$suc]/cuenta/saldohaber)
    let $debe:=sum(/sucursales/sucursal[@codigo=$suc]/cuenta/saldodebe)
return
<sucursal>
    <codigo>{$suc}</codigo>
    {$director}
    {$poblac}
    <suma_debe>{$debe}</suma_debe>
    <suma_haber>{$haber}</suma_haber>
</sucursal>
```

- Devuelve el nombre de los directores, el código de la sucursal, y la población de las sucursales con más de tres cuentas.

```
for $sucursal in /sucursales/sucursal
where (count($sucursal/cuenta) > 3)
return <Sucursal>{($sucursal/@codigo)} {($sucursal/director)}
{($sucursal/poblacion)} </Sucursal>
```

- Devuelve por cada sucursal, el código de sucursal y los datos de las cuentas con más saldo debe

```
for $suc in distinct-values(//sucursales/sucursal/@codigo)
    let $codigo:=//sucursales/sucursal[@codigo=$suc]/@codigo
    let
cuenta:=/sucursales/sucursal[@codigo=$suc]/cuenta[number(saldodebe)=number
(max(//sucursales/sucursal[@codigo=$suc]/cuenta/saldodebe))]

return
<sucursal>
    <codigo>{$suc}</codigo>
    {$cuenta}
</sucursal>
```

- Devuelve la cuenta del tipo pensiones que ha hecho más aportación:

```
for $suc in distinct-values(//sucursales/sucursal/@codigo)
    let
$cuenta:=/sucursales/sucursal[@codigo=$suc]/cuenta[number(aportacion)=numb
er(max(//sucursales/sucursal[@codigo=$suc]/cuenta/aportacion))]
return
<sucursal>
    <codigo>{$suc}</codigo>
    {$cuenta}
</sucursal>
```

Nota: Hay distintas posibles maneras de realizar un mismo ejercicio por lo que puede que existan soluciones correctas distintas de las aquí plasmadas.

¡Buen trabajo!