Pegue aquí el código de barras

"Conforme consta en el acta quinta se expone el cuestionario correspondiente a la prueba realizada el día 15 de junio de 2017 en el Centro Municipal de Formación Permanente (Paseo del Generalife, nº 6, Arroyo de la Miel), correspondiendo a la 2ª prueba de la fase de oposición, incluyendo una respuesta correcta para cada una de las cuestiones realizadas, pudiendo, no obstante, existir en algún caso otras respuestas igualmente válidas dada la naturaleza de la prueba, las cuales, una vez comprobadas que son correctas, se calificarán como tales."

Caso práctico 1:

Puesta en marcha de servidor web con Apache, Mysql y PHP.

En el siguiente caso práctico se va a proceder a la instalación de un servidor web sobre un sistema base con Debian 8 donde estará instalado Apache, Mysql y PHP, es decir, lo que comunmente llamamos servidor LAMP. Este servidor web va a dar servicio sólo a la red local.

Nuestro servidor tendrá la siguiente configuración:

IP Estática: 192.168.1.100/24
Puerta de enlace predeterminada: 192.168.1.1
DNS1: 195.235.113.3
FQDN del servidor: servidorweb.local

Comprobaciones iniciales.

1. ¿Qué comando nos permite visualizar la ip actual del servidor?

Respuesta:

ifconfig

2. Teniendo en cuenta que trabajamos sobre un sistema Debian, ¿en qué fichero podemos ver el resto de los parámetros de red arriba descritos?

a) /etc/init.d/network
b) /etc/eth0
c) /etc/sysconfig
d) /etc/network/interfaces

Respuesta:

d)

Instalación de Apache2.

3. ¿Cuál sería la secuencia correcta para realizar la instalación?

a) sudo apt-get update
sudo apt-get upgrade
sudo apt-get install apache2

b) sudo apt-get remove
sudo apt-get update
sudo apt-get apache2

c) sudo apt-get apache2
sudo apt-get update
sudo apt-get upgrade

Respuesta:

a)

---

4. Una vez realizado el paso anterior si todo ha ido bien, al poner en el navegador:

http://192.168.1.100
nos debe aparecer la página de bienvenida de Apache.
Si queremos, que en vez de poner la ip, podamos poner el FQDN, es decir:

http://servidorweb.local

obteniendo el mismo resultado que con la ip y teniendo en cuenta que no disponemos de un servidor DNS en la red local, ¿qué fichero de sistema tendríamos que configurar si inicialmente durante la instalación del sistema no se ha hecho y la web la estamos abriendo en el navegador del propio servidor?

Respuesta:

/etc/hosts

---

5. Para comprobar que el servicio de Apache se está ejecutando, ¿qué comando podemos ejecutar?

a) ping 192.168.1.100
b) traceroute servidorweb.local
c) nslookup servidorweb.local
d) netstat -ta

Respuesta:

d)

---

6. Vamos a crear un Virtual Host en Apache y para ello creamos un fichero de configuración "miweb.conf".

¿Dónde colocarías este fichero de configuración?

a) /etc
b) /etc/apache2/sites-available
c) /etc/apache2/sites
d) /var/www/html

Respuesta:

b)

---

7. Para habilitarlo en Apache tendremos que ejecutar:

Respuesta:

sudo a2ensite miweb.conf

Instalación de Mysql.

8. Después de la instalación de Mysql, ¿qué script permite mejorar la seguridad de una instalación por defecto de MySQL?

a) sudo mysql -u root -p < secure.sh
b) sudo mysql_secure_installation
c) sudo apt-get install mysql-security
d) Ninguna de las anteriores, no hace falta.

Respuesta:

b)

---

9. Una vez nos hemos logeado en Mysql quereos crear una base de datos "webdata" y dar todos los privelegios (excepto WITH GRANT OPTION) **sobre ésta al usuario "webmaster" con contraseña "mipassword". ¿Cómo lo harías?.**

Respuesta:

create database webdata;
grant all on webdata.* to 'webmasterr' identified by 'mipassword';

NOTA: También es válido como usuario 'webmaster@localhost'

Instalación de PHP.

10. Tras la instalación de PHP en nuestro servidor, queremos comprobar que PHP está funcionando y además conocer qué versión está instalada.
Indique como hacerlo.

Respuesta:
Crear un fichero info.php dentro del directorio Root de nuestro Host Virtual o en el directorio por defecto de Apache con el siguiente contenido:
<?php
Phpinfo();
¿>

Hacer la llamada desde el navegador a la url http://192.168.1.100/miweb/info.php

---

Caso práctico 2:
Javascript

Dados los siguientes códigos javascript responda a las preguntas.

1. Dado el siguiente código:

```
function mifuncion(){
  var total = 2;
  if(total < 3) {
    var total = 4;
    console.log("Valor total:" + total);
  }
console.log("Valor total:" + total);
}
```

Al ejecutar a la función "mifuncion()" ¿Qué resultado se obtendría en la consola?

Valor total: 4
Valor total: 4

---

2. Dado el siguiente código:

```
function mifuncion() {
  let ventas = 2;

  if(ventas < 3) {
    console.log("Es menor que 3");
  }
   console.log("Valor ventas:" + ventas);
}
```

En este código "ventas" sólo se va a asignar una vez, ¿Habría una forma más adecuada de definirla? Especifíquela.

Sí habría una forma más adecuada puesto que si nunca se va a modificar el valor (tan solo cuando se inicializa) es porque queremos una constante y no una variable. Así que sería más apropiado cambiar let por const.

3. **Dado el siguiente código:**

```
var usuarios = ["manuel", "ariadna", "julia",
"pedro"];
// bucle A
for(var i = 0, len = usuarios.length; i < len;
i++)
{
    console.log(usuarios[i]);
}
// bucle B
for(var i = 0; i < usuarios.length; i++)
{
    console.log(usuarios[i]);
}
```

¿Qué diferencias hay entre el bucle A y el B? ¿Generarían el mismo resultado?

Ambos funcionan y devuelven los mismos valores. Pero en el bucle A, el length de los usuarios se comprueba una única vez, mientras que en el bucle B en cada iteración se debe comprobar el length de usuarios, siendo este último menos eficiente.

4. **Dado el siguiente código:**

```
function mifuncion() {
var mi_variable=[];
console.log(mi_variable == false);
console.log(mi_variable === false);
if (mi_variable) console.log('Evaluación ok');
}
```

Al ejecutar a la función "mifuncion()" ¿Qué resultado se obtendría en la consola? Explique brevemente porqué se obtiene dicho resultado.

true
false
Evaluación ok

En javascript existen dos comparaciones == (comparación "relajada") y === (comparación "estricta"). La primera compara el valor mientras que la segunda compara tipo y valor.
Cuando se utiliza la primera forma, si los operandos tienen tipos distintos, javascript trata de realizar una conversión para hacerlos comparables. En el primer caso, en el momento de realizar la comparación de un no-booleano con un booleano, javascript le asocia su valor truthy o falsy equivalente. En el caso de un array vacío, javascript le "asocia" el valor False (por eso la comparación devuelve true).
Con el segundo tipo de comparación, si tienen distintos tipos, directamente se considera que no son iguales (por eso devuelve false).
En el caso de la evaluación del if nos encontramos que javascript, cuando se evalúa un array vacío, lo considera true (de forma distinta a cuando tiene que realizar la comparación "relajada" con un booleano). Por ese motivo se escribiría el mensaje en consola (Evaluación ok).

---

5. **Queremos tener una función genérica que sume las ventas de un array, sin usar librerías de terceros ni funciones propias. Rellene el código que falta.**

```
const ventaSemestre = [20, 30, 40, 43, 23, 10];
const ventaTrimestre = [20, 30, 40];
var sumaVentas = (misventas) =>
misventas.reduce((total, numero) => total + numero)
console.log(sumaVentas(ventaSemestre));
console.log(sumaVentas(ventaTrimestre));
```

La función reduce nos permite convertir un Array de elementos en un único elemento, definiendo la operación que necesitamos para realizar la transformación. El primer parámetro (total) es el resultado, variable que se utiliza para ir almacenando lo que ha sucedido en cada una de las iteraciones, y el segundo parámetro (numero) es el item de cada iteración. De esta forma le estamos diciendo que sobre el array misventas, aplique una transformación que consiste en ir acumulando sobre el parámetro total la suma de cada uno de sus elementos (parámetro numero) y lo devuelva sobre la variable sumaVentas.

Tanto en esta función como en la siguiente hemos usado => denominado arrow function o función flecha. Es una forma sintácticamente más corta de especificar funciones. El operador => está precedido por la lista de argumentos de la función y va seguido por la expresión que se devuelve como resultado de la ejecución de la función.

6. Complete el código que falta para obtener una función genérica que aplique un descuento a las compras que se realicen.

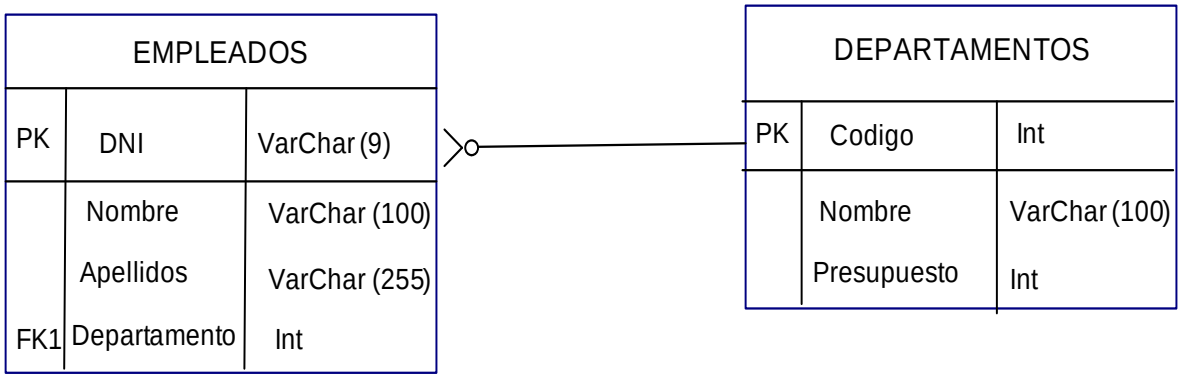
```
const comprasSemestre = [20, 30, 40, 43, 23, 10];
const comprasTrimestre = [20, 30, 40];
var aplicarDescuento = (compras, descuento) =>
compras.map((compra) => compra - (compra *
(descuento/100)));
console.log(aplicarDescuento(comprasSemestre, 10));
console.log(aplicarDescuento(comprasTrimestre, 15));
```

El método map() crea un nuevo array con los resultados de la llamada a la función indicada aplicados a cada uno de sus elementos.
Así cuando realizamos la llamada aplicarDescuento(comprasSemestre, 10) le estamos indicando que aplique un descuento del 10 por ciento sobre comprasSemestre. En la definición de aplicarDescuento usamos arrow function indicando que se llama con dos parámetros: compras y descuento, y que devuelve la expression compras.map((compra) => compra – (compra * (descuento/100))). En esta expresión es dónde usamos map, que tiene un único parámetro: compra, que tomará el valor de cada uno de los elementos del array compras y le aplicará el código restante (devolviendo para cada uno el resultado de evaluar compra – (compra * (descuento/100))

Caso práctico 3:
SQL

En todo lo no contemplado en este supuesto, el aspirante podrá efectuar las suposiciones que considere oportunas, debiendo siempre hacerlas constar en su propuesta de solución con una justificación.

Se decide implementar una base de datos para la gestión de los empleados de una empresa y su ubicación en los distintos departamentos, con el esquema de base de datos que a continuación se detalla:



Escriba las sentencias en ANSI SQL de las siguientes cuestiones:

1. **Obtener todos los datos de los empleados que se apellidan "Vidal" y los que se apellidan "Guerra".**

SELECT * FROM EMPLEADOS WHERE Apellidos='Vidal' OR Apellidos='Guerra';

También sería válido:

SELECT * FROM EMPLEADOS WHERE Apellidos IN ('Vidal','Guerra');

---

2. Obtener todos los datos de los empleados cuyo apellido empiece por "V".

SELECT * FROM EMPLEADOS WHERE Apellidos LIKE 'V%';

Otra opción válida:

SELECT * FROM EMPLEADOS WHERE Apellidos LIKE 'V*';

---

3. Obtener el presupuesto total de todos los departamentos.

SELECT SUM(Presupuesto) FROM DEPARTAMENTOS;

---

4. Obtener el número de los empleados en cada departamento.

SELECT Departamento, COUNT(*) FROM EMPLEADOS GROUP BY Departamento;

---

5. Obtener una lista completa de empleados, incluyendo por cada empleado los datos del empleado y de su departamento.

SELECT * FROM EMPLEADOS INNER JOIN DEPARTAMENTOS ON EMPLEADOS.Departamento = DEPARTAMENTOS.Codigo;

---

6. Obtener los nombre y apellidos de los empleados que trabajan en departamentos cuyo presupuesto sea mayor de 900.000€.

SELECT Nombre, Apellidos FROM EMPLEADOS
WHERE Departamento IN
(SELECT Codigo FROM DEPARTAMENTOS WHERE Presupuesto>900000);

7. Obtener los datos de los departamentos cuyo presupuesto es superior al presupuesto medio de todos los departamentos.

SELECT * FROM DEPARTAMENTOS
WHERE Presupuesto > (SELECT AVG(Presupuesto) FROM DEPARTAMENTOS);

---

8. Añadir un nuevo departamento "Calidad" con presupuesto 70.000 € y código de departamento 4.

INSERT INTO DEPARTAMENTOS VALUES (4,'Calidad',70000);

---

9. Aplicar un recorte presupuestario del 5% a todos los departamentos.

UPDATE DEPARTAMENTOS SET Presupuesto = Presupuesto * 0.95;

---

10. Eliminar a todos los empleados que trabajen para departamentos cuyo presupuesto sea superior a 900.000 €.

DELETE FROM EMPLEADOS WHERE Departamento IN
(SELECT Codigo FROM DEPARTAMENTOS
WHERE Presupuesto>900000);

Caso práctico 4:
Servicios Web / Java

Dada la clase Person.java y la clase PersonUtil.java que detallamos a continuación, hemos implementado un servicio web mediante anotaciones de la api JAX-WS que obtiene una persona usando su Identificador. La solución tiene 3 clases PersonServiceI.java, PersonImpl.java y PersonPublisher.java. El servicio está desplegado en la URL http://www.benalmadna.es/ws. Teniendo en cuenta la situación descrita, responda a las siguientes preguntas sobre el ejercicio planteado:

Person.java:

```
package com.aytoBenalmadena.examen.jaxws;

public class Person {
     private int id;
     private String name;

     public int getId() {
          return id;
     }

     public void setId(int id) {
          this.id = id;
     }

     public String getName() {
          return name;
     }

     public void setName(String name) {
          this.name = name;
     }

     public Person(int id, String name) {
          super();
          this.id = id;
          this.name = name;
     }

     public Person() {
          super();
     }
}
```

PersonUtil.java

```
package com.aytoBenalmadena.examen.jaxws;

import java.util.HashMap;
import java.util.Map;

public class PersonUtil {
     private static Map map;

     static {
          map = new HashMap();
          map.put(1, new Person(1, "A"));
          map.put(2, new Person(2, "B"));
          map.put(3, new Person(3, "C"));
          map.put(4, new Person(4, "D"));
          map.put(5, new Person(5, "E"));
     }

     private PersonUtil() {
     }

     public static Person getPerson(Integer id) {
          return map.get(id);
     }
}
```

PersonServiceI.java

```
package com.aytoBenalmadena.examen.jaxws;

import javax.jws.WebMethod;
import javax.jws.WebService;
//Línea de código Pregunta 1
import javax.jws.soap.SOAPBinding.Style;

//Línea de código 1 Pregunta 2
//Línea de código 2 Pregunta 2
public interface PersonServiceI {
     @WebMethod
     Person getPerson(Integer id);
}
```

PersonServiceImpl.java

```
package com.aytoBenalmadena.examen.jaxws;

import javax.jws.WebService;

// Cadena de declaración de endpointInterface pregunta 3
@WebService(endpointInterface =

"_____________________________________________________")
public class PersonServiceImpl implements PersonServiceI {
```

```
	@Override
	public Person getPerson(Integer id) {

		return PersonUtil.getPerson(id);
	}

}
```

PersonPublisher.java

```
package com.aytoBenalmadena.examen.jaxws;

import javax.xml.ws.Endpoint;

public class PersonPublisher {

	public static void main(String[] args) {
		// Línea de código 1 pregunta 4

		Endpoint ep =
	Endpoint.create(_____________________________________);

		//Línea de código 2 pregunta 4

	ep.publish(_____________________________________________);
	}

}
```

Preguntas:

1) **En la clase PersonServiceI, ¿qué import faltaría por definir?. (Linea de código Pregunta 1).**

   Respuesta:

   import javax.jws.soap.SOAPBinding;

2) **En la clase PersonServiceI, en la declaración de interface, ¿qué atributos JAX-WS hay que declarar?. (Linea de código 1 Pregunta 2 y Linea de código 2 Pregunta 2).**

Respuesta:

@WebService
@SOAPBinding(style = Style.DOCUMENT)

---

3) **En la clase PersonServiceImpl, el atributo @WebService declara la implementación del servicio, ¿qué cadena debe establecerse para la propiedad endpointInterface?(Cadena de declaración de endpointInterface pregunta 3)**

Respuesta:

"com.aytoBenalmadena.examen.jaxws.PersonServiceI"

---

4) **En la clase PersonPublisher, tenemos la creación del Endpoint, ¿Qué parámetro tenemos que pasarle al método create? (//Código 1 pregunta 4)**

Respuesta:

new PersonServiceImpl()

---

5) **En la clase PersonPublisher, después de crear el Endpoint tenemos, que publicar el servicio en la dirección indicada, ¿Cómo sería el código de esta publicación? (Línea de código 2 pregunta 4).**

Respuesta:

ep.publish("http://www.benalmadna.es/ws");

Caso práctico 5:
Hibernate

Queremos usar el framework Hibernate para el acceso a datos de las tablas person, address y person_address que se detallan con las siguientes sentencias SQL de creación de tablas:

```
create table address (
        address_id bigint not null auto_increment,
        buildingName varchar(255),
        postCode varchar(255),
        town varchar(255),
        primary key (address_id)
    );

create table person (
        person_id bigint not null auto_increment,
        firstName varchar(255),
        lastName varchar(255),
        primary key (person_id)
    );

create table person_address (
        person_id bigint not null,
        address_id bigint not null,
        primary key (person_id, address_id)
);
```

Las tablas descritas las vamos a mapear a las siguientes clases de java mediante Hibernate Annotations, donde la tabla person_address representa una relación ManyToMany entre las clases Person y Address:

`Clase Person`:

```
package net.codejava.hibernate.model;
```

```
import java.util.HashSet;
import java.util.Set;

import javax.persistence.CascadeType;
import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.JoinColumn;
import javax.persistence.JoinTable;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "person")
public class Person {

    long id;
    String firstName;
    String lastName;
    Set<Address> addresses;

    public Person() {

    }

    public Person(String firstName, String lastName) {
        this.firstName = firstName;
        this.lastName = lastName;
        this.addresses = new HashSet<Address>();
    }

    @Id
    @GeneratedValue
    @Column(name = "person_id")
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    @Column
    public String getFirstName() {
        return firstName;
    }

    public void setFirstName(String firstName) {
        this.firstName = firstName;
    }

    @Column
    public String getLastName() {
```

```
        return lastName;
    }

    public void setLastName(String lastName) {
        this.lastName = lastName;
    }



    @ManyToMany(cascade = { CascadeType.ALL })

    @JoinTable(name = "person_address",

joinColumns = { @JoinColumn(name =

 "person_id") }, inverseJoinColumns =

{ @JoinColumn(name = "address.id") })
    public Set<Address> getAddresses() {
        return addresses;
    }

    public void setAddresses(Set<Address> addresses) {
        this.addresses = addresses;
    }

}
```

Clase Address

```
package net.codejava.hibernate.model;

import java.util.Set;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.ManyToMany;
import javax.persistence.Table;

@Entity
@Table(name = "address")
public class Address {

    long id;
    String buildingName;
    String town;
    String postCode;
    Set<Person> persons;

    public Address() {

    }
```

```
    public Address(String buildingName, String town, String
postCode) {
        this.buildingName = buildingName;
        this.town = town;
        this.postCode = postCode;
    }

    @Id
    @GeneratedValue
    @Column(name = "address_id")
    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }

    @Column
    public String getBuildingName() {
        return buildingName;
    }

    public void setBuildingName(String buildingName) {
        this.buildingName = buildingName;
    }

    @Column
    public String getTown() {
        return town;
    }

    public void setTown(String town) {
        this.town = town;
    }

    @Column
    public String getPostCode() {
        return postCode;
    }

    public void setPostCode(String postCode) {
        this.postCode = postCode;
    }



    @ManyToMany(mappedBy = “addresses")

    public Set<Person> getPersons() {
        return persons;
    }

    public void setPersons(Set<Person> persons) {
        this.persons = persons;
```

```
    }

}
```

En las clases descritas, se han eliminado los modificadores de los atributos, sustituyéndolos por líneas continuas que deberá rellenar con la cadena adecuada para que el mapeo de las clases esté correctamente declarado.