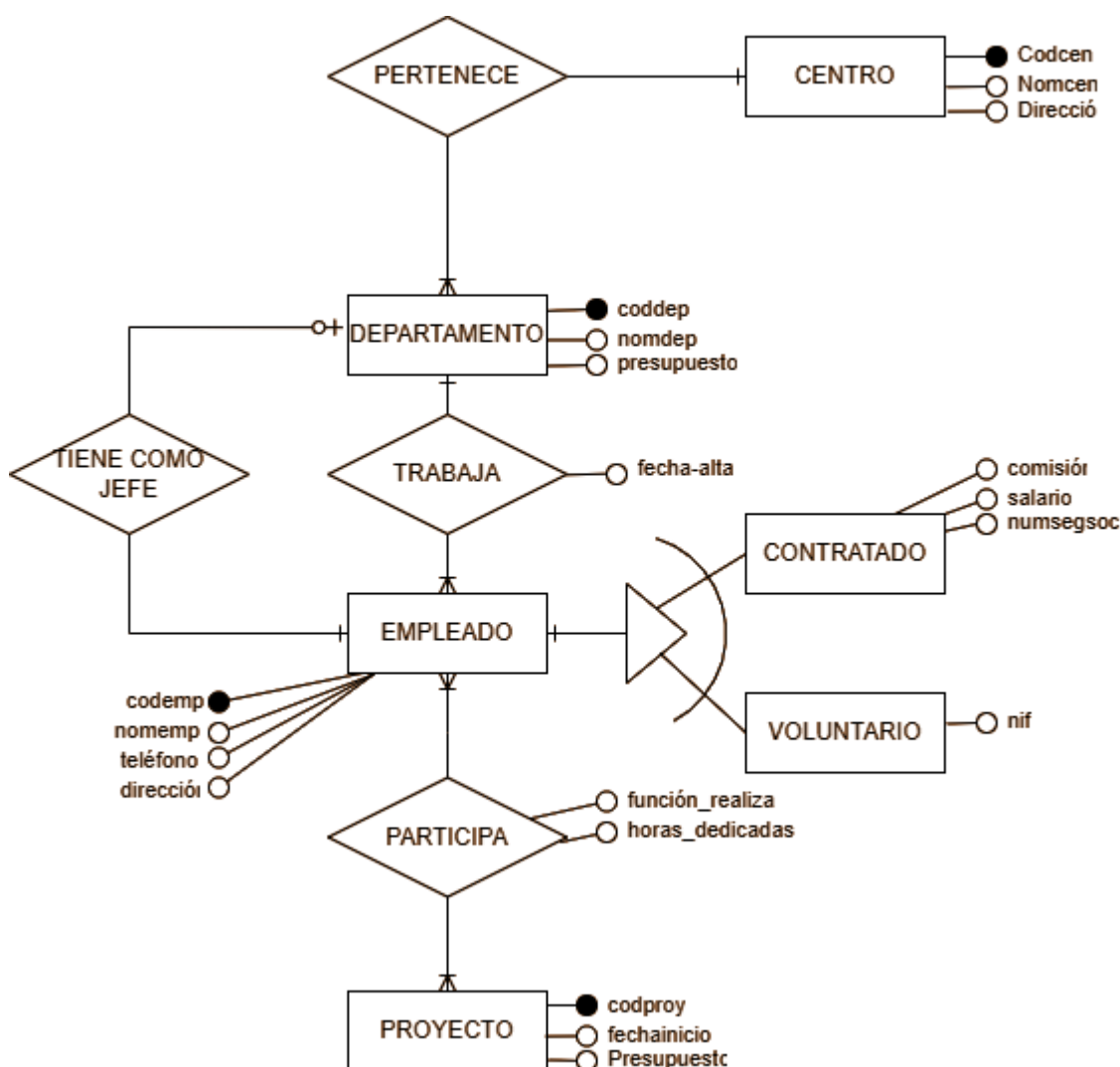


<b>CS-3.1</b>	<b>Enunciado de Prueba</b>	<b>Año:</b>	2024
<b>Especialidad:</b>	INFORMÁTICA		
<b>Prueba</b>	B2. Opción A	<b>Acceso:</b>	1 - 2

### Ejercicio 1

Una empresa desarrolla su actividad en varios centros geográficamente separados. Desea gestionar información sobre los empleados de que dispone y en qué proyectos está trabajando cada uno de ellos. Para ello se ha desarrollado el siguiente modelo entidad relación:



Teniendo en cuenta el diagrama E/R mostrado, se pide:

1. Diseñar el modelo relacional que le corresponde, teniendo en cuenta que:
  - a. Se deben subrayar las claves primarias de cada tabla.
  - b. Se deben indicar claramente las restricciones de integridad referencial (claves ajenas) indicando a qué clave primaria apunta cada una.

2. A partir del modelo relacional diseñado, escribe las sentencias SQL (en Oracle o en MySQL) necesarias para:

- a. Crear la tabla DEPARTAMENTOS con las restricciones de clave primaria y ajena apropiadas, considerando todos los campos obligatorios y con los tipos de datos que consideres más adecuados.

Insertar dos filas en la tabla DEPARTAMENTOS creada en el punto anterior.

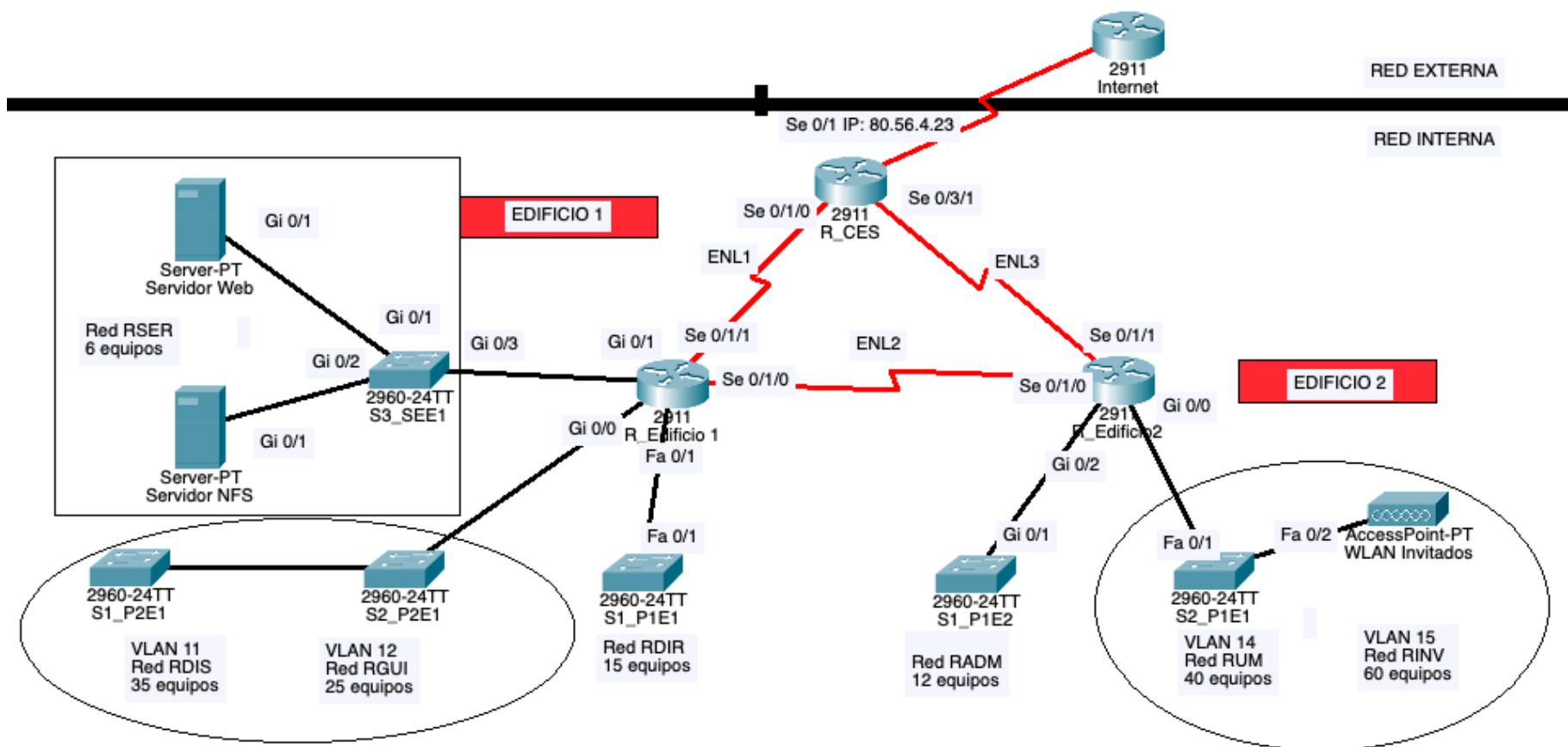
- b. Mostrar el nombre de empleado y salario total (salario más comisión) de aquellos empleados cuyo salario supera al salario medio (sin comisiones) de la empresa. Ordena el resultado de la consulta por salario total de forma descendente.
- c. Mostrar para cada departamento de la empresa su código, nombre, el número de empleados que trabajan en él y la suma de los salarios y sus comisiones, a la que denominaremos *gasto\_dep*. Deberán mostrarse también aquellos departamentos que no tienen empleados. Ordenar la consulta en orden descendente por el gasto del departamento.
- d. Modificar la comisión a todos los empleados del departamento de código 111, para aquellos empleados que trabajan en más de dos proyectos. El nuevo valor de la comisión será el doble del actual.

**Valoración del ejercicio (sobre 10 puntos):**

- *Ejercicio 1: máximo 2 puntos.*
- *Ejercicio 2:*
  - *Apartado a): máximo 2 puntos.*
  - *Apartado b): máximo 2 puntos.*
  - *Apartado c): máximo 2 puntos.*
  - *Apartado d): máximo 2 puntos.*

## Ejercicio 2

La empresa Digit S.L. dispone de una red corporativa ubicada en dos edificios cuya topología se puede observar en la siguiente imagen:



Existen diferentes redes lógicas, con diferentes necesidades de equipos cada una, asociadas a los departamentos de la empresa, en concreto:

- Red *RINV*: Hasta 60 equipos.
- Red *RUM*: Hasta 40 equipos.
- Red *RADM*: Hasta 12 equipos.
- Red *RDIR*: Hasta 15 equipos.
- Red *RGUI*: Hasta 25 equipos.
- Red *RDIS*: Hasta 35 equipos.
- Red *RSER*: Hasta 6 equipos.



Se pide:

1. Partiendo de la dirección IP 172.17.254.0/23 y teniendo en cuenta las necesidades de cada departamento de la empresa, establecer un esquema de direccionamiento VLSM, optimizando el espacio de direcciones tanto como sea posible. Se deben tener en cuenta los enlaces entre routers (en el diagrama de red ENL1, ENL2, ENL3).

Toda la información deberá ser recogida en las siguiente tabla:

Nombre Subred	Dirección Subred	Máscara Subred Decimal/CIDR	Primera Dir. Útil	Última Dir. Útil	Dir. Broadcast
<i>RINV</i>					
<i>RUM</i>					
<i>RDIS</i>					
<i>RGUI</i>					
<i>RDIR</i>					
<i>RADM</i>					
<i>RSER</i>					
<i>ENL1</i>					
<i>ENL2</i>					
<i>ENL3</i>					



2. Asigna direcciones IP a las interfaces de los routers indicados, teniendo en cuenta que debe ser **obligatoriamente la primera IP** disponible en la dirección de subred:

R_Edificio 1		
Interfaz	Dirección IP	Máscara de Subred Decimal/CIDR 255.255.255.255/32
Se 0/1/0 (Primera dirección útil de la subred)		
Se 0/1/1		
Gi 0/0.11 Vlan 11		
Gi 0/0.12 Vlan 12		
Gi 0/1		
Fa 0/1		

R_Edificio 2		
Interfaz	Dirección IP	Máscara de Subred Decimal/CIDR 255.255.255.255/32
Se 0/1/0		
Se 0/1/1		
Gi 0/2		
Gi 0/0.14 Vlan 14		
Gi 0/0.15 Vlan 15		

R_CES		
Interfaz	Dirección IP	Máscara de Subred Decimal/CIDR 255.255.255.255/32
Se 0/3/1 (Primera dirección útil de la subred)		
Se 0/1/0 (Primera dirección útil de la subred)		



3. En toda la red corporativa se utiliza enrutamiento estático. Suponiendo que todos los routers han sido configurados para que la red converja, indica cómo quedarían las tablas de rutas de los routers *R\_Edificio2* y *R\_CES* teniendo en cuenta que:
- En el router *R\_Edificio2*, el tráfico desde las redes *RADM*, *RUM* y *RINV* hacia la red de servidores *RSER* será enviado por el router *R\_Edificio1*. Todo el tráfico restante se enviará por el router *R\_CES*.
  - Se ha utilizado sumarización de rutas (agregación de rutas) siempre que ha sido posible, sin tener en cuenta los enlaces entre routers.

R_Edificio2			
Dir. Red	Máscara/CIDR	Interface	IP Siguiente Salto

*Utiliza solo las filas que sean necesarias*

R_CES			
Dir. Red	Máscara/CIDR	Interface	IP Siguiente Salto

*Utiliza solo las filas que sean necesarias*



4. Los servicios *HTTP/HTTPS* en el *Servidor Web* y el servicio *NFS* en el *Servidor NFS*, ambos en la red *RSER*, deben ser accesibles desde la red pública, por lo que es necesario configurar *tunneling* (reenvío de puertos) en el router *R\_CES*. Además, un usuario administrador debe poder conectarse desde su dirección IP externa (3.4.5.6) por *SSH* a cualquiera de los dos servidores (*Servidor Web* a través del puerto 1000 y al *Servidor NFS* a través del puerto 1001). Teniéndolo en cuenta, indica en la siguiente tabla los parámetros de configuración indicados:

Equipo	Dirección IP/Máscara CIDR
Servidor Web (Última dirección IP útil en la subred)	
Servidor NFS (Penúltima dirección IP útil en la subred)	

R_CES				
IP Externa	Puerto Externo	IP Interna	Puerto Interno	Protocolo

*El protocolo NFS utiliza el puerto 2049 TCP.*

*El protocolo SSH utiliza el puerto 22 TCP.*

*Utiliza solo las filas que sean necesarias.*

- a. ¿Qué comando debería lanzar desde una terminal el usuario *admin* para conectarse al *Servidor NFS* por *SSH* desde su ip externa (3.4.5.6)?.

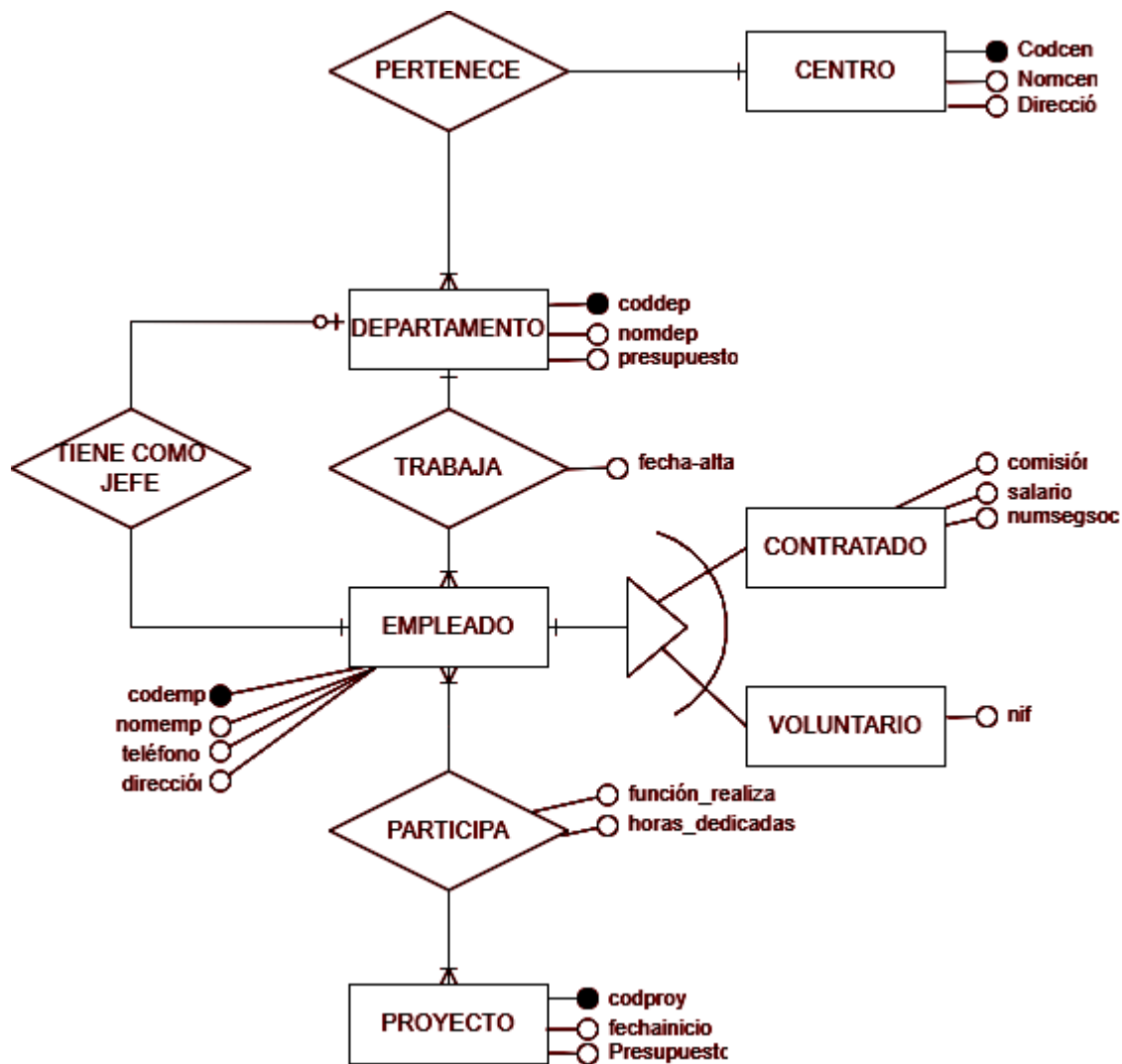
**Valoración del ejercicio (sobre 10 puntos):**

- Apartado 1: Máximo 4 puntos.
- Apartado 2: Máximo 2 puntos.
- Apartado 3: Máximo 3 puntos.
- Apartado 4: Máximo 1 punto.

Prueba	B2. Opción B	Acceso:	1 - 2
--------	--------------	---------	-------

## Ejercicio 1

Una empresa desarrolla su actividad en varios centros geográficamente separados. Desea gestionar información sobre los empleados de que dispone y en qué proyectos está trabajando cada uno de ellos. Para ello se ha desarrollado el siguiente modelo entidad relación:



Teniendo en cuenta el diagrama E/R mostrado, se pide:

3. Diseñar el modelo relacional que le corresponde, teniendo en cuenta que:
  - a. Se deben subrayar las claves primarias de cada tabla.
  - b. Se deben indicar claramente las restricciones de integridad referencial (claves ajenas) indicando a qué clave primaria apunta cada una.





4. A partir del modelo relacional diseñado, escribe las sentencias SQL (en Oracle o en MySQL) necesarias para:
- Crear la tabla DEPARTAMENTOS con las restricciones de clave primaria y ajena apropiadas, considerando todos los campos obligatorios y con los tipos de datos que consideres más adecuados.
  - Insertar dos filas en la tabla DEPARTAMENTOS creada en el punto anterior.
  - Mostrar el nombre de empleado y salario total (salario más comisión) de aquellos empleados cuyo salario supera al salario medio (sin comisiones) de la empresa. Ordena el resultado de la consulta por salario total de forma descendente.
  - Mostrar para cada departamento de la empresa su código, nombre, el número de empleados que trabajan en él y la suma de los salarios y sus comisiones, a la que denominaremos *gasto\_dep*. Deberán mostrarse también aquellos departamentos que no tienen empleados. Ordenar la consulta en orden descendente por el gasto del departamento.
  - Modificar la comisión a todos los empleados del departamento de código 111, para aquellos empleados que trabajan en más de dos proyectos. El nuevo valor de la comisión será el doble del actual.

**Valoración del ejercicio (sobre 10 puntos):**

- *Ejercicio 1: máximo 2 puntos.*
- *Ejercicio 2:*
  - *Apartado a: máximo 2 puntos.*
  - *Apartado b): máximo 2 puntos.*
  - *Apartado c): máximo 2 puntos.*
  - *Apartado d): máximo 2 puntos.*



## Ejercicio 2

Se desea implementar una aplicación para gestionar los trabajadores de una compañía, cuya información se encuentra almacenada en un fichero denominado *trabajadores.csv*. Cada línea del archivo guarda la siguiente información:

*Código del trabajador, nombre, salario, departamento y año de nacimiento (entero de 4 dígitos)*

Un ejemplo del contenido del fichero sería el siguiente:

```
1;Juan Pérez;1500.00;Finanzas;1975
2;María Rodríguez;1800.00;Recursos Humanos;1982
3;Pedro García;1000.00;Producción;1990
4;Gabriel Torres;1200.50;Informática;1989
5;Sofía López;2000.00;Marketing;1985
6;Carlos Díaz;1100.00;Ventas;1992
7;Ana Sánchez;1300.00;Contabilidad;1980
8;Javier Martínez;1600.00;Logística;1987
9;Lucía Gómez;1400.00;Compras;1991
10;Rafael González;1900.00;Dirección;1978
```

**IMPORTANTE:** Tener en cuenta que el fichero *trabajadores.csv* NO tendrá la primera línea como cabecera con el nombre de las columnas.

Se desea cargar en memoria la información de todos los trabajadores para lo que se utilizará una lista genérica *Lista <T>* que dispone de los siguientes métodos:

- *Lista()*: Constructor de la lista que la inicializará vacía.
- *void insertar(T elemento)*: Añadirá un elemento al final de la lista.
- *T obtener(int indice)*: Devolverá el elemento de la posición *índice* que se recibe por parámetro.
- *int tamaño()*: Devolverá el tamaño de la lista, es decir, el número de elementos que contiene.
- *void eliminar(int indice)*: Eliminará el elemento de la posición *índice* que se recibirá por parámetro. Si el índice pasado está fuera del rango de la lista, no eliminará nada.
- *boolean contiene(T dato)*: Devuelve un booleano indicando si el elemento recibido por parámetro está en la lista (*True*) o no (*False*).

Se pide, utilizando como lenguaje de programación **Java o C++**:

1. Implementación de la clase *Trabajador*. Los atributos de la clase serán accedidos a través de los métodos *getters* y *setters* correspondientes.
2. Implementación de la clase *TrabajadoresCSV* que tendrá los siguientes métodos para trabajar con la *lista*, denominada *listaTrab*, que tendrá como atributo de clase:
  - a. *metodo1*: Carga la información de los trabajadores en la lista *listaTrab*. Recibe como parámetro una cadena con el nombre del fichero a cargar.



- b. *metodo2*: Retorna una lista con los trabajadores que pertenezcan a un departamento cuyo nombre es recibido por parámetro.
- c. *metodo3*: Retorna el valor medio de los salarios de los trabajadores de la lista que hayan nacido entre los 2 años recibidos por parámetro, ambos incluidos.
- d. *metodo4*: Retorna una lista que contendrá los trabajadores cuyo año de nacimiento fue bisiesto.

Un año es bisiesto:

- Si el año es divisible por 400.
  - Si el año es divisible por 4 y no por 100.
- e. *metodo5*: Volcará en un archivo de texto los trabajadores de la lista que pertenezcan a un departamento cuyo nombre es recibido por parámetro. El archivo se llamará “*trabajadores\_dpto\_XXX.csv*” donde XXX será el nombre del departamento requerido. Se deberá mostrar al final el mayor salario de los trabajadores de dicha lista y el nombre del trabajador que lo tiene (Se puede suponer que sólo habrá 1 trabajador con el salario máximo).

**Valoración del ejercicio (sobre 10 puntos):**

- *Ejercicio 1: Máximo 0.75 puntos.*
- *Ejercicio 2:*
  - *Apartado a): Máximo 2 puntos.*
  - *Apartado b): Máximo 1.5 puntos.*
  - *Apartado c): Máximo 1.5 puntos.*
  - *Apartado d): Máximo 2 puntos.*
  - *Apartado e): Máximo 2.25 puntos.*