


```
#ID: E098
#NOMBRE: Elena Martínez
#NIVEL: senior
#CARGO: Arquitecto de Software
#DEPARTAMENTO: Innovación
#FECHA CONTRATACIÓN: 2015-04-22 (9 AÑOS CONTRATADO)
####FIN MOSTRAR EMPLEADOS
```

- **CP1.2.** Cuando se introduzca un nivel en el que haya empleados, se mostrarán esos empleados:

```
#### <tuNombre> MUESTRA EMPLEADOS CON FILTRO: junior
## EMPLEADO NÚMERO 1
#ID: E032
#NOMBRE: Roberto García
#NIVEL: junior
#CARGO: Analista
#DEPARTAMENTO: Consultoría
#FECHA CONTRATACIÓN: 2021-03-15 (3 AÑOS CONTRATADO)
## EMPLEADO NÚMERO 2
#ID: E037
#NOMBRE: Marcos Fernández
#NIVEL: junior
#CARGO: Desarrollador
#DEPARTAMENTO: Tecnología
#FECHA CONTRATACIÓN: 2022-01-10 (2 AÑOS CONTRATADO)
####FIN MOSTRAR EMPLEADOS
```

- **CP1.3.** Cuando se introduzca un nivel del que NO haya empleados, se mostrará lo siguiente:

```
#### <tuNombre> MUESTRA EMPLEADOS CON FILTRO: principiante
#NO HAY EMPLEADOS EN ESE NIVEL
####FIN MOSTRAR EMPLEADOS
```

- **CP1.4.** Cuando se introduzca un valor nulo en división, no se analizará el fichero, y se mostrará lo siguiente:

```
#### EL FILTRO NO PUEDE SER NULO. NO SE ANALIZA EL FICHERO.
```

2 Encontrar empleado.

Implementa un manejador para que encuentre un empleado en la empresa, y que muestre nombre, cargo, nivel y departamento. Requisitos y casos de prueba.

- Admitirá una cadena de texto como parámetro, que será el id del empleado.
- Se buscará al empleado ignorando mayúsculas y minúsculas.
- **CP2.1.** Encuentra empleado. Cuando lo encuentra deberá dejar de analizar el archivo.

```
#### <tuNombre> BUSCA A e087:
```

```
## Lucía Sánchez es Gerente de proyecto senior, dentro del  
departamento de Gestión.
```

```
#### FIN BUSCAR EMPLEADO.
```

- **CP2.2.** No encuentra al empleado. Informará de que ese empleado no está contratado en la empresa.

```
#### <tuNombre> BUSCA A E100:
```

```
## No se encuentra al empleado con identificador E100.
```

```
#### FIN BUSCAR EMPLEADO.
```

- **CP2.3.** Si el ID de empleado no está informado o contiene una cadena vacía, no se analizará el fichero.

```
#### Para buscar a un empleado debes informar el ID.
```

3 Revisar IDs de empleado

Realiza un programa que lleve a cabo la búsqueda de empleados en un XML con SAX. Los empleados que buscará el programa estarán guardados en un fichero de texto plano CSV con dos campos. Por ejemplo (existe ya este archivo con el nombre `entrada.csv`):

```
E041;Ana López
E087;José Manuel
E100;Roberto García
E037;Luis Erans
E098;Juan Carlos Fresneda
E120
E120;jose;Jefe
```

El programa leerá cada una de las líneas del fichero de texto plano, y realizará una búsqueda SAX por cada línea (buscará cada empleado en el XML). Si lo encuentra, y es incorrecto, lo grabará correctamente en otro fichero CSV llamado “correcciones.csv” con los datos `ID;Nombre;Cargo;Telefono`. Requisitos y casos de prueba:

- **CP3.1.** La salida normal será la siguiente, cuando en el fichero plano existan empleados:

```
#### <tuNombre> BUSCA EMPLEADOS POR FICHERO:
# E041 es Ana López (Desarrollador)
# E087 no es José Manuel, si no Lucía Sánchez (Gerente de
proyecto)
# E100 (Roberto García) no es empleado de la empresa.
# E098 no es Juan Carlos Fresneda, si no Elena Martínez
(Arquitecto de software)
# FORMATO INVALIDO. LAS LÍNEAS DE BÚSQUEDA DEBEN TENER 2
COLUMNAS.
# FORMATO INVALIDO. LAS LÍNEAS DE BÚSQUEDA DEBEN TENER 2
COLUMNAS.
#### FIN BUSCAR EMPLEADOS POR FICHERO.
```

Se grabará el archivo `correcciones.csv`:

```
E087;Lucía Sánchez;Gerente de Proyecto;+34 987 654 321
E098;Elena Martínez;Arquitecto de Software;
```

- **CP3.2.** El fichero plano está vacío (`entradaVacia.csv`)

```
#### EL FICHERO entradaVacia.txt ESTÁ VACÍO.
```

- **CP3.3.** El fichero plano no existe (cualquier nombre que no exista).

```
#### EL FICHERO NO EXISTE
```