Desenvolvemento de aplicacións multiplataforma
MP0483 Examen sistemas informáticos 3ª eval
Fecha: 7 Junio/2022
Nombre y apellidos:
Dni:

VALORES DE LAS PREGUNTAS
Ejercicio 1: deploy: 7 ptos
Ejercicio 2: shell script: 3 ptos

Ejercicio 1: Deploy en servidores

Deploya en un servidor el siguiente proyecto web (usando el framework Spring Boot)

La aplicación web tendrá que tener las siguientes funcionalidades:

1-
Cuando un cliente acceda a la dirección del endpoint:
http://localhost:8080/letradni/**XXXX**

Nos devolverá la **letra del dni** que se corresponde con el número del dni pasado como parámetro XXXX

Pista: es fácil calcular la letra de un dni dado
https://www.geekno.com/pasos-para-calcular-la-letra-del-dni-con-java.html

Validación: Tienes que validar el número de dni introducido. En caso de que no te introduzcan un dni válido (8 dígitos enteros), devolveremos un mensaje de error

Ejemplos:
http://localhost:8080/letradni/36335432 → D
http://localhost:8080/letradni/36335 → Error
http://localhost:8080/letradni/alejando → Error

2-
Cuando un cliente acceda a la dirección del endpoint:
*http://localhost:8080/coincidencias?**nombre1=XXXX**&**nombre2=YYYY***

Nos devolverá una página web con el total de letras coincidentes entre XXXX e YYYY.
Ej si XXXX= Pepito e YYYY=Pepa, nos debe devolver 2, ya que la letra 'P' y la 'e' están repetidas en los dos nombres. No se distinguirá entre mayúsculas y minúsculas.

Ejemplos:
*http://localhost:8080/coincidencias?**nombre1=pepito**&**nombre2=Pepa*** → ***2***
*http://localhost:8080/coincidencias?**nombre1=P**&**nombre2=pepito*** → ***1***
*http://localhost:8080/coincidencias?**nombre1=M**&**nombre2=pepito*** → ***0***

3-
Ahora se pide un endpoint que dado un país o ciudad, mostremos por pantalla el número de habitantes que posee.
http://localhost:8080/habitantes/**Vigo**

Para ello usaremos el **API** gratuito de geolocación https://open-meteo.com
En concreto su endpoint:
https://geocoding-api.open-meteo.com/v1/search?name=**Vigo**

NOTA: este api no devuelve un solo valor, sino un conjunto de ellos (un array), nosotros nos vamos a mostrar solo el primero de ellos.

Ejemplos:
http://localhost:8080/habitantes/**Vigo** → **297332**
http://localhost:8080/habitantes/**London** → **7556900**
http://localhost:8080/habitantes/**Spain** → **46723749**

4-
Por último se pide un endpoint que permita guardar en una base de datos el nombre de un alumno y su nota.

Deberás usar (para la construcción del ejercicio) un servidor de base de datos (PostgreSQL) y tendrá que estar montado en un servidor (Rayway o Heroku).

El método para guardar la información lo decides tú.

Recuerda:

- Aplicación web **pusheada** en un repositorio **privado** de **github** del alumno
- **Despliegue** de la aplicación realizado correctamente en servidor **Heroku o Railway** del alumno
- El programa tendrá que compilar y funcionar sin errores

Puntuaciones del ejercicio:

- Funcionalidad 1 del servidor backend realizada correctamente (dni): 2 pto
- Funcionalidad 1 del servidor backend realizada correctamente (coincidencia): 1 pto
- Funcionalidad 1 del servidor backend realizada correctamente (api población): 2 ptos
- Funcionalidad 1 del servidor backend realizada correctamente (BD): 2 ptos

Ejercicio 2: Shell Script

Crea un programa en shell script (linux) que pida constantemente por pantalla valores numéricos por teclado.
El programa dejará de pedir valores cuando el usuario introduzca el valor 0 en la terminal

- Si el valor insertado es negativo:
    - Creará una carpeta llamada **datos**
    - Creará (dentro de la carpeta datos) un fichero llamado **examen.txt** cuyo contenido será el valor numérico (negativo) insertado por teclado, dividido por 2
        - (tendrás que hacer los cálculos con 2 decimales)
    - Si el directorio datos ya existiese lo borrará (junto con todo su contenido) (mostrando el mensaje "La carpeta examen ya existía y la he borrado"
- Si el valor insertado es cero:
    - El programa terminará, mostrando el mensaje "FIN"
- Si el valor insertado es positivo
    - Mostrará por pantalla todos los valores comprendidos entre ese número y 1 (es decir recorrer de forma inversa …,5,4,3,2,1)

Recuerda que el programa tendrá que volver a pedir de nuevo otro número, y volver a comenzar el bucle desde el principio.
Tendrás que usar **funciones** obligatoriamente (por claridad en el código).
El programa tendrá que compilar y funcionar sin errores

Puntuaciones del ejercicio:
- Funcionalidad 1 guardar valor negativo dividido entre 2: 1 pto - Funcionalidad 2 terminar programa cuando insertamos 0: 1 pto - Funcionalidad 3 mostrar números del X a 1: 1 pto