

Explicación Práctica (REST)

Antes de nada, he hecho un video en el que muestro el funcionamiento de la práctica, la ejecuto para diversos casos, y además, explico de forma general las distintas partes del código, que hace cada una, etc... El video esta subido a youtube por tanto no hay que descargárselo:

<https://www.youtube.com/watch?v=D-2VDg84RQo&feature=youtu.be>

Descripción del Cliente (Parte 1)

Para la parte del cliente se pide consumir un servicio externo (API MetaWeather). Esta API nos va a proporcionar la previsión del tiempo para los siguientes 5 días (además de otra información) en forma de JSON, por tanto se trata de obtener esa información mediante una petición GET, procesar la información, obtener la que nos interesa, analizarla, y en base a ella elaborar una recomendación de viaje.

Lo primero que hago, para no tener que introducir directamente el WOEID de la ciudad que queremos consultar, es un método que a partir del nombre de la ciudad, realiza una petición para consultar su WOEID (la propia API permite hacer esta consulta).

```
public static String getWOEID (String ciudad)
```

Una vez que ya obtengo el WOEID, realizo el GET a la información que interesa (la previsión). A partir del JSON que proporciona, proceso la información y obtengo la que necesito (abreviaturas del tiempo, temperatura máxima y mínima).

```
public static String getRecomendacion (String JSON)
```

Con esa información, calculo cuantos días lloverán en los siguientes 5 días (dependiendo de la abreviatura que se obtiene, si es un tipo de lluvia, incremento un contador)

```
public static int analizarTiempo (String[] abreviaturas)
```

Además, calculo la temperatura media (en mi caso primero he calculado la media de todas las mínimas, luego la de todas las máximas, y entre esas dos otra media, que es la que uso para determinar el tipo de ropa que hay que llevar (manga larga o corta)

```
public static int mediaTemperatura (int[] min, int[] max)
```

A partir de toda esta serie de consultas y procesamiento de la información, se llega de la previsión del tiempo obtenida a la recomendación que corresponde. He puesto el nombre de cada método por si quieres ir consultando el código, que te sea más fácil.

Descripción del Servicio (Parte 2)

Para la segunda parte, se pide desarrollar una API REST que se encargue de realizar 2 tareas. Una es almacenar todas las recomendaciones que se obtienen a partir de las consultas que se hagan (junto con la fecha y hora y el WOEID).

```
public static String getFecha()
```

La otra tarea es poder recuperar las recomendaciones almacenadas según el WOEID de una ciudad concreta, todo mediante intercambio con JSON.

Para la tarea de almacenar las recomendaciones, he usado el verbo POST, que crea un nuevo recurso, que en este caso será la recomendación construida en el cliente. Cada vez que se recibe un nuevo recurso, lo almaceno en un fichero llamado *reco.txt*.

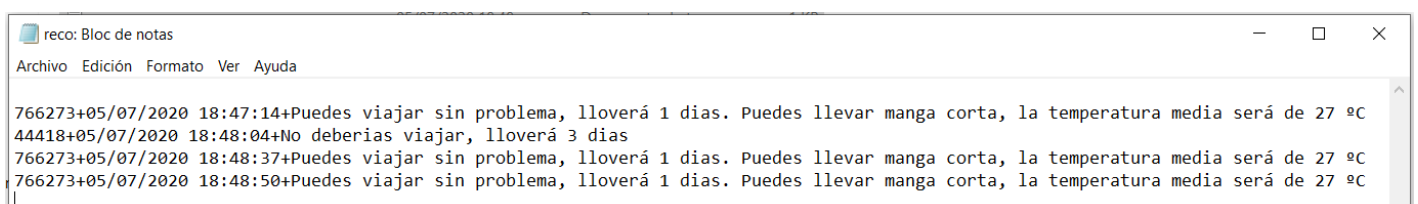
```
@POST
@Path("/json")
@Produces(MediaType.APPLICATION_JSON)
@Consumes(MediaType.APPLICATION_JSON)

public static Recomendacion guardarRecomendacion(
    @QueryParam("fecha") String fecha,
    @QueryParam("recomendacion") String recomendacion,
    @QueryParam("WOEID") String WOEID
) throws IOException
```

El formato que he decidido usar para almacenar la información en el *.txt*, es usar una línea para cada recomendación, separando el WOEID, la fecha y la propia recomendación con el carácter “+”.

Quizás era mas conveniente almacenar la información directamente en forma de JSON en un archivo *.json* o *.txt*, pero finalmente decidí hacerlo así porque si se hacen muchas recomendaciones, se optimiza el uso de los caracteres, y es incluso mas legible que JSON (aunque este ya es de por si muy fácil de leer).

Por ejemplo, el fichero *reco.txt* quedaría de la siguiente forma tras almacenar unas cuantas recomendaciones:



```
reco: Bloc de notas
Archivo Edición Formato Ver Ayuda

766273+05/07/2020 18:47:14+Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 27 ºC
44418+05/07/2020 18:48:04+No deberias viajar, lloverá 3 dias
766273+05/07/2020 18:48:37+Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 27 ºC
766273+05/07/2020 18:48:50+Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 27 ºC
```

En mi caso, las peticiones POST las realizo desde mi Cliente desarrollado en la primera parte, aunque podrían hacerse desde otro cliente que proporcione la información necesaria

```
public static void postRecomendacion(String recomendacion, String WOEID)
```

Para la recuperación de las recomendaciones para una ciudad concreta, hago uso de un GET, que a partir del WOEID, busca en el fichero *reco.txt* todas las recomendaciones coincidentes.

```
@GET
@Path("/json")
@Produces(MediaType.APPLICATION_JSON)

public String getList(@QueryParam("WOEID") String WOEID){
    return leerArchivo(WOEID);
}
```

El método *leerArchivo()* es el que se encarga de buscar dentro del archivo, y “construir” la lista de todas las recomendaciones obtenidas a partir del WOEID.

Por ejemplo, si se quiere consultar unas recomendaciones para Madrid que se hayan hecho con anterioridad, se acabaría mostrando lo siguiente al usuario:

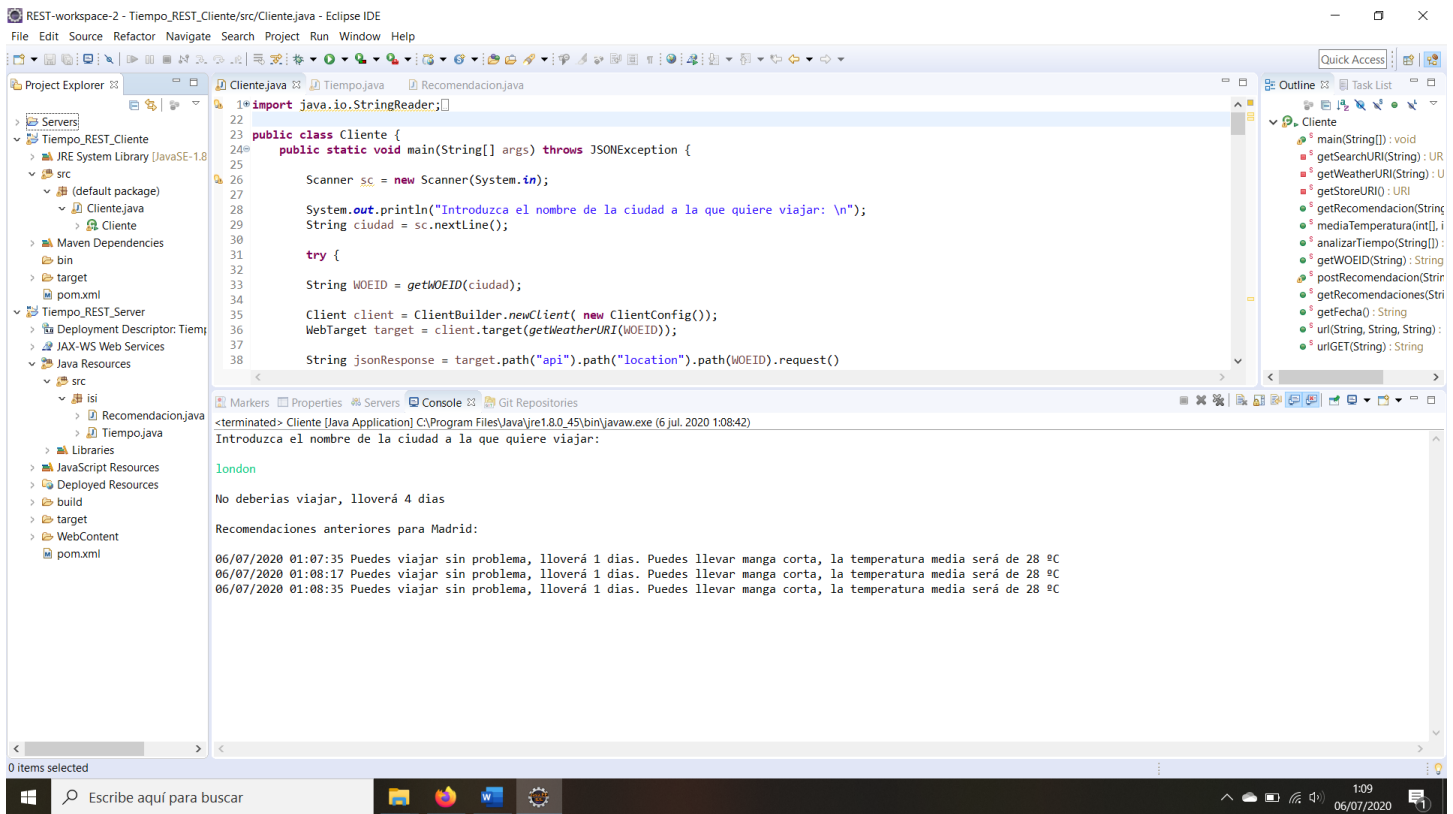
```
05/07/2020 18:47:14 Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 27 ºC
05/07/2020 18:48:37 Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 27 ºC
05/07/2020 18:48:50 Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 27 ºC
```

Evidencias del funcionamiento

Como no se si esta descripción de la práctica es correcta o es demasiado escueta, he decidido hacer un video explicativo como indico al principio del *pdf*, en el que enseño el funcionamiento y voy narrando las distintas funciones. Espero que pueda servir de complemento a las explicaciones y como evidencia de funcionamiento:

<https://www.youtube.com/watch?v=D-2VDg84RQo&feature=youtu.be>

Aun así, como en las indicaciones de la práctica solo se mencionan las capturas, voy a incluir una en la que hago una consulta de Londres, y además recupero las recomendaciones hechas anteriormente para Madrid:



```
1 import java.io.StringReader;
22
23 public class Cliente {
24     public static void main(String[] args) throws JSONException {
25
26         Scanner sc = new Scanner(System.in);
27
28         System.out.println("Introduzca el nombre de la ciudad a la que quiere viajar: \n");
29         String ciudad = sc.nextLine();
30
31         try {
32
33             String WOEID = getWOEID(ciudad);
34
35             Client client = ClientBuilder.newClient( new ClientConfig());
36             WebTarget target = client.target(getWeatherURI(WOEID));
37
38             String jsonResponse = target.path("api").path("location").path(WOEID).request()
```

<terminated> Cliente [Java Application] C:\Program Files\Java\jre1.8.0_45\bin\javaw.exe (6 jul. 2020 1:08:42)

Introduzca el nombre de la ciudad a la que quiere viajar:

london

No deberias viajar, lloverá 4 dias

Recomendaciones anteriores para Madrid:

06/07/2020 01:07:35 Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 28 ºC

06/07/2020 01:08:17 Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 28 ºC

06/07/2020 01:08:35 Puedes viajar sin problema, lloverá 1 dias. Puedes llevar manga corta, la temperatura media será de 28 ºC

Victor Miguel Mora Alcazar

Practica 3 (Laboratorio de Integración de Sistemas Informáticos)

Curso 2019/2020