PROYECTO LEX: AEMET PARSER

MODELOS DE COMPUTACIÓN

Francisco Miguel Toledo Aguilera y Alejandro Manuel Hernández Recio

3º Grado en Ingeniería Informática – Curso 2017/2018

ugr | Universidad de Granada

TABLA DE CONTENIDO

Tabla de contenido

1. Introducción

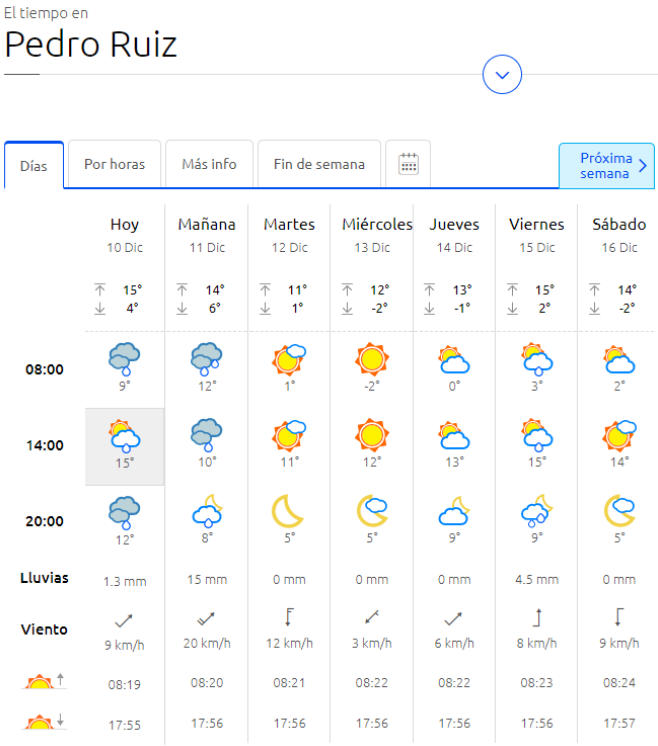
ESCENARIO

Hemos diseñado un programa que, dinámicamente y en función de una localidad seleccionada, descarga de AEMET un archivo con la predicción meteorológica de dicha localidad. Ese fichero es interpretado mediante Lex para generar una página web con los datos recogidos.

PROBLEMAS DURANTE EL DESARROLLO

Intentamos usar una librería para hacer las peticiones HTTP, pero dejó de funcionar y no pudimos arreglar ese fallo.

Nuestra primera idea era que la página generada fuese similar a la de ElTiempo.es



Pero cuando estábamos trabajando sobre el fichero de predicción, nos dimos cuenta de que faltaban datos en algunos lugares. Donde debería haber un entero indicando la velocidad del viento (por ejemplo), no había nada (“”). Cuantos más días avanzaba la predicción, menos datos aparecían, e incluso algunas etiquetas las omitían y daban directamente el valor, sin saber qué estábamos leyendo.

Es por eso que, finalmente, optamos por suprimir la parte de conexión HTTP e interpretar directamente un fichero pre-descargado, arreglado donde faltaban datos, y que contenía sólo un día.

2. Nuestro programa

FUNCIONAMIENTO

El programa al iniciarse parsea el fichero de localidades, y carga en memoria las provincias con sus respectivas localidades. Se imprime por pantalla una lista de provincias para que el usuario seleccione una. Cuando la selecciona, carga las localidades de esa provincia y las muestra por pantalla para que el usuario vuelva a elegir una.

Con el código de la provincia (dos dígitos) y el código de la localidad (tres dígitos), se forma el ID para hacer la consulta meteorológica. Tras hacer la consulta, se parsea el fichero de la predicción y se extraen los datos para generar la web.

EJECUCIÓN

Compilar con *make* y ejecutar *prediccion* sin pasarle argumentos. La web generada está en *index.html*. Abrir con el navegador preferido.

FICHERO LOCALIDADES.JSON

Este fichero se tuvo que crear para poder hacer las peticiones HTTP, ya que cada localidad está identificada por un ID, que no es igual a su código postal. En principio queríamos hacer una consulta a una web para que el usuario introdujese el nombre de una localidad, y se buscase su código postal para hacer la consulta. Después nos dimos cuenta de que el CP no tenía por qué coincidir con el ID de la localidad.

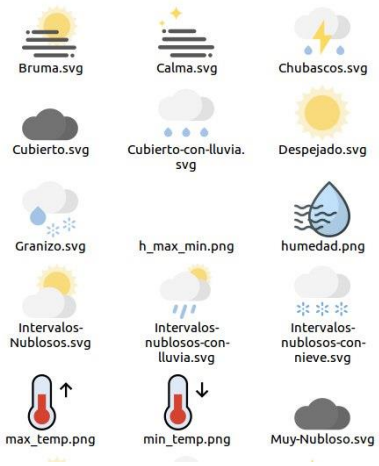
Desde AEMET se proporciona un CSV con todos los códigos de provincias y localidades, así que tuvimos que editarlo a mano hasta dejarlo en formato JSON. El incluido con el proyecto es uno de prueba y está incompleto, pues sólo contiene tres provincias con cuatro o cinco localidades cada una.

FICHERO PREDICCIÓN.JSON

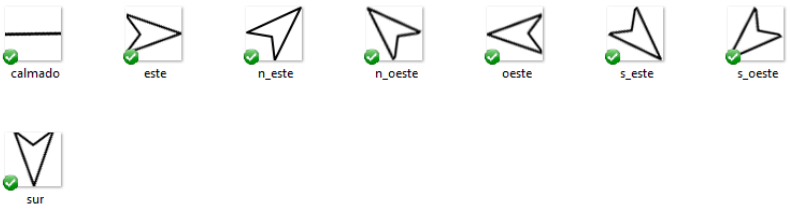
Es en este fichero donde se encuentra el meollo de la cuestión, pues es el centro de nuestro programa. Como hemos mencionado antes, se interpreta con Lex y se extraen los datos que queramos a unas de las estructuras de datos que hemos diseñado para albergarlas, detalladas a continuación:

- ***struct datos_viento***. Este struct contiene un string para la dirección del viento, y un int para la velocidad del mismo.
- ***struct datos_temp_humedad_sensacion.*** Como temperatura, humedad y sensación térmica tienen los mismos atributos (un máximo, un mínimo, y un map de horas (int) y grados (int)), decidimos crear un struct igual para los tres. Está compuesto de dos int (máxima y mínima), y un map<int hora, int grados>.
- ***map<int, string> desc_estados_cielo***. Este map nos sirve para identificar un estado del cielo con una imagen, para cuando generamos la página web:

```
(0, "Despejado"));
(11, "Despejado"));
(12, "Poco-Nubloso"));
(13, "Intervalos-Nublosos"));
(14, "Nubloso"));
(15, "Muy-Nubloso"));
(16, "Cubierto"));
(17, "Nubes-Altas"));
(18, "Intervalos-nublosos-con-lluvia"));
(19, "Nubloso-con-lluvia"));
(20, "Muy-nubloso-con-lluvia"));
(21, "Cubierto-con-lluvia"));
(22, "Intervalos-nublosos-con-nieve"));
(23, "Nubloso-con-nieve"));
(24, "Muy-nubloso-con-nieve"));
(25, "Chubascos"));
(26, "Tormenta"));
(27, "Granizo"));
(28, "Bruma"));
(29, "Niebla"));
(30, "Calma"));
```



- ***map<string, int> prob_precipitacion.*** Aquí guardamos las probabilidades de precipitación por periodos. Los periodos van de seis en seis horas, y están en el fichero predicción como 00-06, 06-12, 12-18 y 18-24.
- ***map<string, int> estados_cielo.*** Este es el análogo al anterior, pero guardamos los estados del cielo. Con estos estados, luego accedemos a desc_estados_cielo para obtener las imágenes.
- ***map<string, datos_viento> viento***. Al igual que los dos anteriores, guarda el período para la predicción (00-06, 06-12,…) junto al struct de los datos del viento.
- ***map<string, string> desc_direcc_viento:*** al igual que con el estado del cielo, aquí recuperamos la descripción de la dirección del viento, para el icono correspondiente en la web.
-

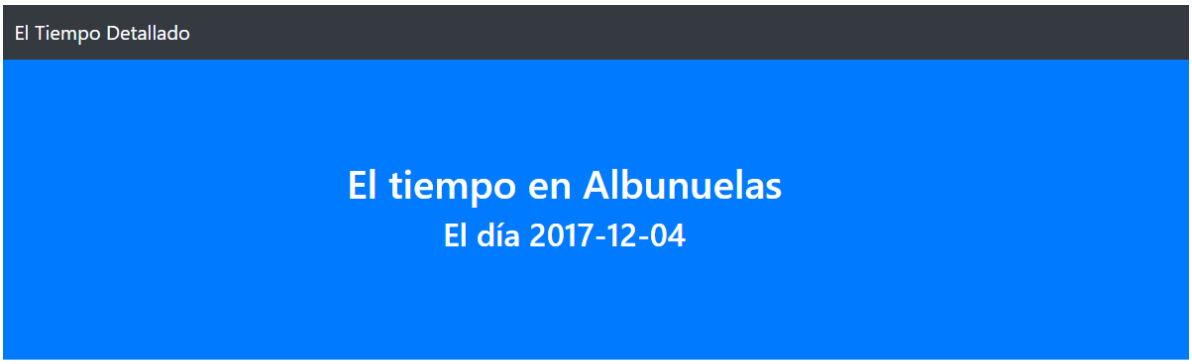



```
("C", "calmado"));
("S", "sur"));
("N", "norte"));
("E", "este"));
("O", "oeste"));
("NE", "n_este"));
("SE", "s_este"));
("SO", "s_oeste"));
("NO", "n_oeste"));
```

GENERACIÓN DE LA WEB

En el main generamos el fichero html llamado "index.html" donde generamos una vista de los datos parseados en predicción.json:

1º parte donde mostramos el nombre de la localidad y la fecha para consultar el tiempo detallado:



2º parte donde mostramos detalladamente y por franja de horas, el estado del cielo, los grados en cada franja, etc…