**CASO PRÁCTICO**

Nos han solicitado una aplicación web para consultar la previsión climatológica en cualquier municipio de España. Después de estudiar las distintas formas de obtener esta información, se ha determinado que lo mejor sería recuperarla vía API de AEMET. Para esto deberemos:

* Solicitar una API Key desde <https://opendata.aemet.es/centrodedescargas/inicio>.
* Revisar la [documentación de la API](https://opendata.aemet.es/dist/index.html) y buscar los métodos necesarios para recuperar la siguiente información. En concreto tendremos que localizar los siguientes servicios:
* Recuperar un listado con todos los municipios de España
* Recuperar la predicción específica para un municipio

**BACKEND**

Se deberá implementar un backend usando **Spring Boot** que exponga una capa de servicios REST con las siguientes operaciones:

* **Búsqueda de municipios por nombre**

Este servicio realizará una llamada a la API de AEMET para listar todos los municipios y se quedará con aquellos que cumplen con los criterios de búsqueda.

* **Obtener predicción para el día siguiente**:

Este servicio realizará una llamada al API de AEMET para obtener la previsión del día siguiente al actual, devolviendo los datos necesarios que se usarán por el frontend (ver en la sección del frontend).

*DATOS DE ENTRADA: El servicio debe recibir los siguientes parámetros:*

* *Código de municipio (obtenido mediante el servicio anterior)*
* *Unidad de temperatura: Sistema de medición de la temperatura a aplicar. Posibles valores: G\_CEL (Grados Celsius) / G\_FAH (Grados* Fahrenheit*)*

*(\*) El parámetro que indica la Unidad de temperatura será opcional, tomando el valor G\_CEL en caso de que no se especifique.*

*DATOS DE SALIDA: Objeto JSON con los siguientes campos:*

* *Temperatura media: Temperatura media medida en la unidad expresada en el parámetro de entrada.*
* *Unidad de medida de temperatura: Indica la unidad utilizada (G\_CEL / G\_FAH)*
* *Probabilidad de precipitación: Array de objetos JSON con el desglose horario de probabilidad de precipitación.*

***Ejemplo de respuesta de nuestro servicio:***

{  
 "mediaTemperatura": 66.7,  
 "unidadTemperatura": "G\_FAH",  
 "probPrecipitacion": [  
 {  
 "probabilidad": 5,  
 "periodo": "00-06"  
 },  
 {  
 "probabilidad": 10,  
 "periodo": "06-12"  
 },  
 {  
 "probabilidad": 10,  
 "periodo": "12-18"  
 },  
 {  
 "probabilidad": 5,  
 "periodo": "18-24"  
 }  
 ]  
}

**FRONTEND**

Se implementará un frontal usando la **última versión de Angular** en la que se presenten 2 campos de entrada:

* Autocompletado con el municipio. Puedes usar cualquier librería para implementar este componente. Por ejemplo [Angular Material](https://material.angular.io/components/autocomplete/overview). Este componente debe llamar al servicio de búsqueda de municipios que has implementado en el backend.
* Unidades de medida de temperatura (Opcional). Por defecto puede aparecer vacío.

Cualquier cambio en los datos introducidos por el usuario en estos 2 campos debe provocar un refresco de la interfaz de usuario.

A modo de referencia, se muestra un ejemplo de cómo podría quedar la interfaz.



**Consejos sobre la implementación:**

* No te compliques con lo bonito que quede a nivel de interfaz. Lo importante es que utilices las buenas prácticas a la hora de organizar el código y los componentes.
* Ten en cuenta que esta primera versión de la aplicación es un MVP (Producto Mínimo Viable) con el que se saldrá al mercado para ver la respuesta de los usuarios. Puesto que se espera que la respuesta sea buena, deberíamos considerar que en futuras iteraciones la funcionalidad se irá evolucionando. Se valorará por lo tanto una correcta separación en capas que permita una buena extensibilidad y mantenibilidad de la solución.

**ENTREGA**

* La entrega de la prueba debe realizarse en un repositorio público de Github. Deberás facilitarnos el enlace a dicho repositorio cuando la tengas acabada.
* Genera imágenes Docker para el frontend y el backed y súbelas a un container registry que se accesible públicamente.
* Deberás entregar las instrucciones para la ejecución en local de la aplicación que has construido Eso no lo he miraod aun, como n

**PREGUNTAS**

***Responde a las siguientes cuestiones y háznoslas llegar una vez acabes la prueba. Te volveremos a contactar para que nos cuentes la solución implementada y comentaremos también este apartado***

* Habrás observado que las llamadas al servicio para obtener el listado de Municipios son bastante lentas. ¿Cómo haríamos para mejorar el tiempo de respuesta de este servicio?
* Hacer una paginación de resultados, en lugar de devolver todos los municipios en una sola respuesta, el servicio podría proporcionar un conjunto limitado de municipios por página.
* Implementar un sistema de caché para almacenar el resultado de las llamadas anteriores.
* Optimizar las consultas. Se puede mejorar en cómo está estructurada la llamada o como se ejecuta las consultas.
* Notarás además que el servicio para recuperar la predicción para un municipio devuelve una URL a la que hacer la consulta. ¿Por qué motivo piensas que AEMET ha implementado este servicio así?
* Los datos meteorológicos cambian con frecuencia.
* Por el tamaño de los datos que se manejan en las consultas.
* Ofrece flexibilidad y escalabilidad.
* Ofrece control de acceso y seguridad.
* A nuestros usuarios les ha gustado la aplicación, pero les parece muy engorroso tener que indicar cada vez que acceden a la aplicación el municipio donde se encuentran para consultar la previsión del tiempo. Les gustaría que se recordara el último municipio seleccionado y que automáticamente les cargue la predicción del día siguiente. ¿Cómo lo podríamos hacer?
* Persistencia en Base de Datos.
* Uso de Cookies.
* Usar el almacenamiento en local en el cliente.
* La aplicación está teniendo tanto éxito que estamos causando una carga excesiva en el servicio de AEMET, vulnerando la siguiente cláusula:

*El usuario se compromete a utilizar esta página o las diferentes formas de acceso, sin incurrir en actividades que puedan ser consideradas ilícitas o ilegales, que infrinjan los derechos de AEMET o de terceros, o que puedan dañar, inutilizar, sobrecargar o deteriorar el sitio web u otras formas de acceso o impedir la normal utilización del mismo.*

Tras la amenaza de AEMET con cortarnos el servicio, debemos apresurarnos a buscar alguna solución que afecte lo mínimo posible a nuestros usuarios. ¿Qué podríamos hacer? ¿Cómo implementarías la solución?

* Usar un limitante de consultas por cliente.
* CPD físico + Cloud con nuestro servicio para escalarlo horizontalmente.
* Utilizar un sistema de cola para gestionar las consultas al servicio de AEMET. En lugar de realizar las consultas de manera inmediata, se encolan las solicitudes y se procesan gradualmente para evitar una carga excesiva en el servicio. Esto permitirá controlar la velocidad de las consultas y reducir la carga en el servidor de AEMET.

**POSIBLES MEJORAS**

* Flujo de Integración continua: La cuenta personal de Github que has creado dispone de capacidad para automatizar la ejecución de pruebas de forma gratuita. Realiza la configuración necesaria para automatizar estas pruebas.

|  |
| --- |
| ¡**No te quedes bloqueado!** En caso de que necesites ayuda escríbenos un email y te contestaremos lo antes posible  [jvegara@grupoasv.com](mailto:jvegara@grupoasv.com)  [jarubio@grupoasv.com](mailto:jarubio@grupoasv.com) |