Prueba técnica en Spring.

Este documento presenta una prueba técnica destinada a evaluar las habilidades y conocimientos de los candidatos en el desarrollo de Java, con un enfoque específico en el uso de Spring Framework.

La prueba técnica abarca una variedad de temas, desde los principios básicos de Java y los conceptos esenciales de Spring hasta desafíos más complejos relacionados con la manipulación de datos, la seguridad y la integración de servicios. Al evaluar a los candidatos a través de esta prueba, buscamos no solo medir su competencia técnica, sino también su capacidad para resolver problemas de manera creativa y su habilidad para aplicar los conocimientos teóricos en situaciones prácticas del mundo real sin olvidar la importancia de las decisiones que se tomen en cuenta para publicar nuestras aplicaciones en una arquitectura moderna, corporativa y escalable como lo es microservicios.

A través de esta evaluación, buscamos identificar a los profesionales que no solo pueden desarrollar aplicaciones Java robustas y eficientes, sino que también tienen la capacidad de innovar y mejorar continuamente, lo que es esencial en un campo tan dinámico como el desarrollo de software. La prueba técnica que sigue está cuidadosamente diseñada para ayudarnos a encontrar a los individuos más calificados y apasionados que se unirán a nuestro equipo, contribuirán significativamente a nuestros proyectos y se convertirán en parte integral de nuestra visión tecnológica futura.

Pre-requisitos

Es recomendable contar con los siguientes insumos para ejecutar exitosamente esta prueba.

1. Conocimientos en Java y el framework de desarrollo Spring.
2. Conocimiento en contenerización de aplicaciones y conocimiento de Kubernetes.
3. Poseer una cuenta personal en Github (https://github.com)
4. Poseer una cuenta personal en Docker (https://www.docker.com)
5. Poseer una cuenta personal en OpenWeather (https://openweathermap.org)
6. Poseer una cuenta personal en Auth0 (https://auth0.com)
7. Windows/Mac/Linux
8. Java, OpenJDK 17
9. Gradle
10. IDE de su preferencia para desarrollo en Java
11. Contenerización (puede usar Docker Desktop)
12. Kubernetes (puede usar la instancia con Docker Desktop)
13. Herramienta de diagramación (puede usar Draw.io - https://app.diagrams.net)

Entregables esperados

1. Se le solicitar acceso al repositorio en GitHub con las fuentes que contengan la solución al enunciado.
2. Se le solicitar versionar su archivo Dockerfile

3. Se le solicita el nombre y tag de su imagen en el repositorio público de Docker.
4. Se le solicita comparta los datos de integración con el proveedor de seguridad:
    a. Compartir ClientID
    b. Compartir ClientSecret
    c. Compartir Audiencia
    d. Compartir URL de servidor de autorizaciones
5. Se le solicita comparta los datos de integración con OpenWeather:
    a. Compartir el API KEY
6. Se le solicitar versionar los archivos YAML que permitan la publicación en Kubernetes.

Opcionales (se tomarán en cuenta como un plus en la evaluación)

1. Se le solicita los diagramas solicitados como parte del enunciado.
2. Se le solicita listar las buenas prácticas y/o patrones que utiliza en su desarrollo de la solución.
3. Se le solicita que comparta recomendaciones que vea conveniente para el despliegue de su solución en ambientes nube utilizando Microsoft Azure.
4. Se le solicita listar los puntos que usted implementaría para mejorar o completar su solución y por limitaciones de tiempo, conocimiento, recursos no logró hacerlo como parte de esta prueba.
5. Se le solicita listar o implementar puntos que haya detectado que debe mejorarse como parte del enunciado que usted considere que no debe ser implementado en un ambiente empresarial.

Puntos para evaluar

- Entregables mínimos solicitados.
- Uso de patrones y buenas prácticas incorporadas.
- Documentación que considere necesaria que complementa sus entregables.
- Nivel de completitud de su prueba.

Enunciado

1. Por favor, verifique y lea todo el enunciado antes de comenzar.
2. Construya un API tipo REST utilizando Spring Boot versión 3.1.4, cuyo propósito será abstraer la implementación de un proveedor externo (OpenWeatherMap.org) permitiendo consultar las condiciones del tiempo actuales para una latitud y una longitud específica.
3. Crear el proyecto Spring Boot usando Gradle Groovy
4. Package name será com.pfcti.weather
5. Lenguaje de programación será Java 17 utilizando el OpenJDK, puede utilizar la distribución de su elección.
6. Formato exclusivo será JSON (application/json)
7. Las bitácoras serán configuradas mediante logback, para el alcance de esta prueba utilizando el ConsoleAppender y hacer uso de Slf4j.
8. Todo error debe agregarse como parte de la bitácora.
9. La estructura del proyecto se deja a su elección.
10. Debe incorporar OpenAPI v3 para los recursos web donde se refleje el nombre y la descripción; y para las operaciones de cada recurso web donde se refleje el id, summary, descripción, definición de parámetros y posibles respuestas para los códigos que se indican en enunciados posteriores.
11. Documente su código cuando lo que considere necesario.
12. En caso de omisión o ingreso de parámetros incorrectos retornar un error status 400 Bad Request
13. En caso de ser una solicitud sin autorización retornar un error status 401 Unauthorized
14. En caso de que datos o que recursos no se encuentren retornar un status 404 Not Found
15. EN caso de excepciones no controladas retornar un status 503 Server Error
16. Todos los puntos anteriores con la siguiente estructura:

```
{
  "code": "<<STATUS>>",
  "errors": [
    "<<Detalle del error>>",
  ]
}
```

17. Crear un recurso llamado /api/v1/weather que responde a un HTTP Verb POST para el siguiente payload

```
{
  "lat": 10.01,
  "lon": -84.10
}
Donde lat es un número que representa una latitud y lon un número que representa
la longitud y ambos son requeridos.
```

18. Realizar la integración del API público de OpenWeatherMap.org con el fin de obtener las condiciones climatológicas actuales según la geolocalización de un cliente. La referencia la puede encontrar en https://openweathermap.org/current

19. Basado en los parámetros de ingreso al recurso /api/v1/weather se deben tomar los datos de entrada y hacer el llamado hacia el API de OpenWeatherMap.org con el fin de poder recuperar los datos deseados.
20. Implemente un base de datos H2 en memoria con una tabla llamada WeatherHistory donde los criterios principales de consulta son por lat y por la lon, agregue los siguientes 5 campos adicionales: weather, tempMin, tempMax, humidity, created

```
Donde weather es una cadena de texto que va a ser la condición actual del clima,
tempMin y tempMax son ambos numéricos y corresponde a los límites de temperatura
y el humity un elemento también numérico que marca el porcentaje de humedad en
ese momento. Created es la fecha de registro del registro.
Recuerde que lat y lon son números que representan latitudes y longitudes.
```

21. Cada vez que se dé una solicitud de consulta del tiempo para una lat y lon se debe revisar previamente si esta existe en la base de datos en memoria, y siempre y cuando la diferencia entre la fecha de creación de ese registro y la fecha de consulta actual sea menor a 10 minutos se retorna el valor desde la base de datos; caso contrario se consulta el API del tercero y se ingresa o actualiza el valor en la base de datos.
22. Para el recurso /api/v1/weather, en caso de ser válido y sin errores en el momento de la atención, se debe retornar un JSON con código 200 y el siguiente contenido:

```
{
  "weather": "Cloud",
  "tempMin": 295.16,
  "tempMax": 297,
  "humidity": 83,
}

Donde weather es una cadena de texto que va a ser la condición actual del clima, tempMin
y tempMax son ambos numéricos y corresponde a los límites de temperatura y el humity un
elemento también numérico que marca el porcentaje de humedad en ese momento.
```

23. Crear un recurso llamado /api/v1/weather/history que responde a un HTTP Verb GET,donde se retorne todo el contenido de la base de datos en memoria que el servicio contenga. La estructura de respuesta usted la elige.
24. Adicione seguridad a su API de tipo OAuth2 usando el client_credentials Flow, convirtiendo su API en un Resource Server. Al habilitar la seguridad cada uno de los endpoints anteriores requiere un encabezado de Authorization para funcionar.
25. Su Authorization server debe ser de acceso público, puede generar una aplicación Machine-to-Machine en Auth0 para un API en https://www.auth0.com utilizando el algoritmo de firmado RS256 y Grant Type Client Credentials. Si prefiere otro diferente a Auth0 puede hacerlo sin problema.
26. Será necesario tener a la mano el URL del servidor, el client Id, el secreto, scopes y audiencias para la revisión.
27. Cree un recurso /sys/status que pueda ser invocado de forma anónima, no requiere autorización previa que debe retornar un código 200 únicamente.
28. Genere la imagen de su API (contenerización)
29. Suba su imagen a docker.com de forma pública utilizando el tag que usted defina.
30. Debe compartirse la imagen para la revisión.

31. Genere los yml requeridos para desplegar la imagen como servicio dentro de un kubernetes local que cumpla con las siguientes condiciones:
    - Seleccione el namespace de su preferencia
    - Uso de deployment
    - Disponibilidad por service
    - 1 sola réplica
    - Escalamiento automático basado en RAM a su criterio a 2 réplicas máximo
32. Versionar todas las fuentes necesarias en su espacio de GitHub. En el caso de ser privado se compartirá el GitHub del personal de la organización los cuales requieren acceso para su revisión
33. Realice un diagrama que permita entender su aplicación con el nivel de detalle que usted vea conveniente para su presentación, puede ser subido a su GitHub como parte de su Readme.