
BCNC
CONSULTING GROUP

CÓMO LEER EL DOCUMENTO:

- Este documento tiene como objeto servir como guía de apoyo para realizar el test proporcionado.
- Se presentan los temas que consideramos que pueden generar más dudas.
- Existe un apartado en el que se explica cómo realizar la planificación del test así como comentarios a tener en cuenta. Esto es muy importante ya que nos ayudará a comprender el test en su totalidad y a no dejarnos ningún requisito por el camino.
- Hemos incluido en un apartado los criterios de evaluación que sabemos que el cliente tiene en cuenta a la hora de realizar la evaluación del test.
- También se encuentra otro apartado en este documento dedicado a enlaces que pueden ayudar en la resolución de dudas sobre cómo abordar algunos de los retos técnicos que se presentan en el test.
- Ante cualquier duda, por favor contactad con vuestro contacto de HR en BCNC y os solventarán las dudas pertinentes.

BCNC
CONSULTING GROUP

JAVA DEVELOPER TEST GUIDANCE

PLANIFICACIÓN Y COMENTARIOS:

1. Lo primero que debemos hacer es leer con detenimiento el test. Una vez leído con calma, dibujaremos en papel o digitalmente la estructura de capas que vamos a desarrollar, teniendo en cuenta que tenemos una base de datos y hay que llevar esos datos hasta el controller del API.

2. A continuación, generaremos la estructura de carpetas y archivos del proyecto en nuestro IDE favorito.

3. Añadir un README al repositorio es altamente recomendable, ya que así estaremos aportando profesionalidad al proyecto, por pequeño que sea. Lo utilizaremos para explicar lo que se ha hecho en el test y cómo utilizarlo, cómo correr las pruebas, cómo correr el proyecto, patrones utilizados, etc.

4. En este paso se recomienda la generación de la tabla en la base de datos H2, por ejemplo de manera automática al start.

5. A partir de aquí aconsejamos empezar a programar teniendo en cuenta lo que ya conozcáis y en caso de duda revisar los links que proporcionamos, entre otros.

6. Hay muchas formas de realizar este test, desde más compleja hasta las más sencillas. Nuestra recomendación es que se aproveche para aplicar los principios SOLID y Clean Code, así como manejo de interfaces y patrones de diseño de software.

7. Una vez terminado el test, revisar punto por punto las directrices del mismo para asegurarnos de que cumplimos con todo lo requerido.

8. Como comentarios adicionales:

    1. Realizar el código lo más limpio, modular y escalable posible.

    2. Aportar comentarios en las partes del código donde pueda haber confusión, variables autodescriptivas, etc.

    3. Toda mejora es bienvenida, así que en caso de realizar mejoras, habría que documentarlas en el README, explicándolas y justificando el porqué se ha realizado.

CRITERIOS DE EVALUACIÓN:

1. Claridad del código, se tiene en cuenta lo fácil que resulte leerlo y entenderlo.
2. Un buen readme, explicando tecnologías, arquitecturas y patrones utilizados, además de otras cosas que debe tener un readme.
3. Número de alertas que tenga cuando se ejecute un linter, como por ejemplo Sonar.
4. Separación en capas y que cada capa contenga únicamente su responsabilidad.
5. Uso correcto de la arquitectura elegida para la desarrollar la solución, se prefiere hexagonal, aunque se valora mejor realizar una arquitectura que se conozca y se maneje bien que usar hexagonal y que no esté bien implementada.
6. Se valorará el desarrollo orientado a la eficiencia, no sólo debe funcionar, sino que las consultas a las bases de datos y/o algoritmia utilizados deben ser lo más eficientes posibles.
7. Uso de las herramientas que proporciona Github o similares para la gestión del código. Uso de ramas, tags, etc.
8. Que pasen correctamente todos los tests que se piden en la prueba y al menos un tests de integración para ver el desarrollo del mismo.

NOTA: Cualquier aclaración/explicación/justificación añadidla en el README, nos gusta leer el criterio utilizado.

H2 Database:

- Quickstart -> http://www.h2database.com/html/quickstart.html

SpringBoot API REST:

- https://spring.io/guides/tutorials/rest/

README:

- Ejemplo de un readme completo y estructurado:
  - https://github.com/openfin/java-example/blob/master/README.md
  - https://bulldogjob.com/readme/how-to-write-a-good-readme-for-your-github-project

Patrones de diseño:

- Para poder consultar y elegir patrones de diseño -> https://refactoring.guru/es/design-patterns/java